

Full length article

TRIPLE: A blockchain-based digital twin framework for cyber–physical systems security

Sabah Suhail^{a,*}, Mubashar Iqbal^b, Rasheed Hussain^c, Saif Ur Rehman Malik^{d,e}, Raja Jurdak^f

^a Queen's University Belfast, Belfast, United Kingdom

^b University of Tartu, Tartu, Estonia

^c University of Bristol, Bristol, United Kingdom

^d COMSATS University Islamabad, Islamabad, Pakistan

^e Trinity College Dublin, The University of Dublin, Ireland

^f Queensland University of Technology, Brisbane, Australia

ARTICLE INFO

Keywords:

Blockchain

Cyber–physical system (CPS)

Digital twin (DT)

Threat intelligence (TI)

Threat hunting

ABSTRACT

Cyber–physical systems (CPSs) are being increasingly adopted for industrial applications, yet they involve a dynamic threat landscape that requires CPSs to adapt to emerging threats during their operation. Recently, digital twin (DT) technology (which refers to a virtual representation of a product, process, or environment) has emerged as a suitable candidate to address the security challenges faced by dynamic CPSs. DT has the capability of strengthening the security of CPSs by continuously mapping the physical to twin counterparts to detect inconsistencies. The existing DT-based security solutions are constrained by untrustworthy data dissemination as well as limited data sharing among the involved stakeholders, which, in turn, limit the ability of DTs to run accurate simulations or make valid decisions. To address these challenges, this paper proposes a modular framework called TRusted and Intelligent cyber-PhysicalL systEm (TRIPLE), that leverages blockchain, DTs, and threat intelligence (TI) to secure CPSs. The blockchain-based DT components in the framework provide data integrity, traceability, and availability for trusted DTs. Furthermore, to accurately and comprehensively model system states, the framework envisions fusing process knowledge for modeling DTs from system specification-based and learning-based information and other sources, including infrastructure-as-code (IaC) and knowledge base (KB). The framework also integrates TI for future-proofing against emerging threats, such that threats can be detected either reactively by mapping the behavior of physical and virtual spaces or proactively by TI and threat hunting. We demonstrate the viability of the framework through a proof of concept. Finally, we formally verify the TRIPLE framework to demonstrate its correctness and effectiveness in enhancing CPS security.

1. Introduction

Integrating information and operational technology (IT and OT) systems in cyber–physical systems (CPSs) provides promising solutions in a plethora of ubiquitous industrial ecosystems [1]. This integration also introduces novel attack vectors, such as man-in-the-middle (MitM) attacks [2,3]. Usually, in such systems, operational functionality outweighs security [4]. Furthermore, if addressed at all, security is often added retrospectively during the operation phase of the system rather than at the initial design phase [5]. Thus, loopholes in the system infrastructure enable attackers to gain deeper knowledge of the system behavior and launch advanced persistent threats (APTs). Considering human–machine collaboration scenarios, such attacks may also pose human safety risks in addition to the disruption of services. One such

recent example of an attack tailored for an industrial control system (ICS) is Industroyer2 [6]–a follow-up attack of Industroyer (2016) [7] that targeted the Ukrainian energy company to cause power outages. Considering specifically designed covert and long-term attacks that access and exfiltrate information from systems, essential measures to fortify CPS involve: (1) operational behavior evaluation of the system; and (2) security posture assessment of the system to identify vulnerabilities or threats. To this end, assessing the system security of live and operational CPSs calls for online solutions that accurately reflect the actual CPS in operation while avoiding any interference or implications of testing on the live system [8]. In this regard, traditional behavior-based intrusion detection systems (IDSs) can uncover malicious activities by identifying deviations or patterns from a defined benign behavior [9].

* Corresponding author.

E-mail address: s.suhail@qub.ac.uk (S. Suhail).

<https://doi.org/10.1016/j.jii.2024.100706>

Received 25 July 2023; Received in revised form 22 August 2024; Accepted 16 October 2024

Available online 24 October 2024

2452-414X/© 2024 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

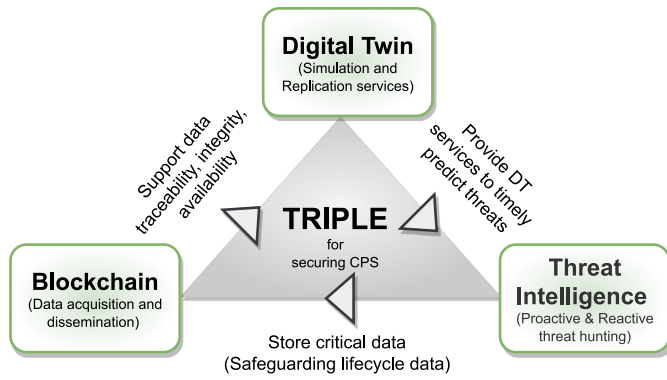


Fig. 1. TRIPLE framework: A triad of blockchain, digital twin, and threat intelligence.

Another potential solution involves deploying testbeds as a controllable cyber prototype for system testing [10]. However, such approaches are rendered either insufficient or infeasible because they are prone to false alarms or alert fatigue, and are unable to reproduce attack scenarios. Additionally, testbed setup and maintenance can be time-consuming and cost-intensive, and they do not cover the entire system's lifecycle, ranging from engineering to operation phase [8,11].

To address the above-mentioned issues, a solution is needed with the following characteristics: (i) it must analyze and optimize the physical processes within CPS before their real-world implementation, (ii) it must perform system testing while linking the entire system's lifecycle to observe the cascading effects, and finally, (iii) it must prevent, detect, and mitigate security loopholes in the live CPS without obstructing the ongoing operations [8]. Fundamentally, digital twin (DT) technology is the most suitable and natural candidate to fulfill the above-mentioned requirements due to the following reasons: DT allows the creation of a digital replica of the underlying CPS where every facet of the CPS can be replicated along with its features, services, and connections. Furthermore, DT allows a feedback loop between the physical CPS and its digital counterpart such that data flows from CPS to the twin while informed decisions on the data, predictions, adjustments, and calibration-related information flow from the twin counterpart to the live CPS.

Leveraging DTs to secure CPS has been discussed by several existing works (see Table 1). However, these studies lack a thorough consideration of design and operational requirements, i.e., process knowledge, data acquisition and integration, context-aware and scalable instantiation, and services necessary to model and integrate building blocks of DTs [12]. Moreover, for information security, physical and virtual entities must use trustworthy data sources, models, and rules. Therefore, a DT framework, at a granular level, must be formulated to consider aspects related to trustworthy data, DT modeling, and integration of DT building blocks to secure CPS.

1.1. Contribution

To address the aforementioned limitations of the traditional approaches, we refine our scope by considering works that (a) focus on industrial use cases, and (b) use DTs as a security-enhancing platform with or without blockchain. Table 1 summarizes the existing works that focus on using DTs, with or without blockchain, while highlighting the essential and/or desirable features¹ necessary for modeling DTs. In this work, we propose a modular framework called **TR**usted and **I**ntelligent cyber-**P**hysical **s**ystem (**TRIPLE**) that leverages the triplet of blockchain, DTs, and threat intelligence (TI) to secure CPSs (as Fig. 1

shows). The TRIPLE framework is a modular framework allowing for flexible implementation based on (a) specific use cases, (b) objectives, and (c) available resources. Since it comprises blockchain, DTs, and TI, it gives CPS system operators (for instance manufacturing plant operators) flexibility and a choice to adopt the features critical for specific use cases within the remit of CPS. For instance, it may support different types of blockchains, i.e., permissionless, public, and private permissioned, DTs with different characteristics [13], and TI to detect threats proactively and/or reactively.

Following the security-by-design approach, the TRIPLE framework ensures data reliability and traceability during data acquisition and dissemination through blockchain. Unlike the traditional approaches that rely either on specification-based or learning-based approaches for building process knowledge of DTs, i.e., modeling of DTs, TRIPLE proposes combining traditional approaches with infrastructure-as-code (IaC) to instantiate comprehensive and scalable DTs. Furthermore, TRIPLE also proposes the concept of combining TI with DT security operation modes (replication and simulation) to predict overarching threats or attacks. Our main contributions are as follows.

- We propose a modular TRIPLE framework (Section 3) that leverages and integrates a provenance-enabled blockchain solution with DT to ensure the reliability and traceability of data during data acquisition and dissemination in both physical and virtual environments.
- We propose two concepts for enhancing security: (a) 'security through DT' which involves utilizing DT as an enabler for intrusion detection to safeguard CPS, and (b) 'security of DT' which entails integrating blockchain technology with DTs to securely store critical data from both CPS and DTs. To achieve comprehensive and accurate modeling of DTs, we propose integrating learning-based and specification-based models with the IaC approach for building process knowledge. We also propose combining DT security operation modes (simulation and replication) with TI to detect the threats proactively and reactively. Reactive threat hunting could be used during time-dependent data synchronization, a module of replication mode, between virtual and physical entities whereas proactive threat hunting, could be used for timely indication of presumable ongoing attacks.
- The proposed TRIPLE framework is modular and can be adapted for any industrial manufacturing scenarios characterized by static production configurations, where the equipment, processes, and workflows remain fixed and do not change frequently. However, without loss of generality, we demonstrate the applicability of the TRIPLE framework through a preliminary proof of concept (Section 4) by using the automotive assembly line as a use case. We provide the implementation of DT model, integration of blockchain, and DT services. Furthermore, we formally model and verify (Section 5) the TRIPLE framework.

1.2. Paper organization

The rest of the paper is organized as follows. Section 2 covers necessary background information and related work. Section 3 proposes the TRIPLE framework by presenting the requirements for a DT framework to secure CPS. Section 4 demonstrates a proof of concept implementation. Section 5 formally models and analyzes the proposed TRIPLE framework. Section 6 provides a discussion of our findings. Finally, Section 7 concludes the paper. The glossary in Appendix C explains key terms related to DTs.

2. Background and related work

This section introduces the preliminary terms necessary for understanding the background of this work. It then provides an overview of existing literature and identifies research gaps (Section 2.5).

¹ The challenges, including Challenge#1, Challenge#2, and Challenge#3 are further discussed in Section 3.

2.1. Digital Twins (DTs)

Driven by asset-centric data, DTs are the virtual (digital) representation of physical systems that mirror every facet of a product, process, or environment, and can provide actionable insights through monitoring, optimization, and prediction [14]. During industrial operations, DT integrates data from multiple sources [14], such as sensors and actuators which are installed in the physical environment, domain knowledge acquired from experts/engineers, manuals, contextual knowledge, system historical lifecycle data [2], and other DT instances or reference DTs [12]. Based on the process knowledge, i.e., modeling of DTs acquired through real-time and historical data, DT follows the states of its physical counterpart. Following a closed loop, DT inspects for data inconsistencies between the physical entity and its virtual counterpart and feeds back the optimized parameters to the physical entity to adopt better calibration and security-enhancing strategies.

2.2. Significance of integrating blockchain in DT

The system's lifecycle and corresponding DT span different lifecycle phases, ranging from engineering to operation [2]. During these phases, multiple industrial stakeholders such as engineers, operators, security professionals, or analysts carry out various operations on the physical device or process [3]. These operations may include update commands to programmable logic controllers (PLCs) and time-dependent modification to the process variables [15]. Other such operations include updating DT data such as user-specified simulation parameters, and cyclic state updates. Due to the presence of cybersecurity threats, it is imperative to guarantee the trustworthiness of data (and sources of data), in both physical systems and their DTs, that are subjected to access privileges through the aforementioned system's lifecycle. Any changes to data must be guaranteed through data immutability and traceability, as the accumulated data underpin rational decision-making [3]. Additionally, the integrity and availability of essential data across multiple stakeholders are critical to ensure accurate state updates between physical and virtual entities. In this regard, blockchain-based DTs can be leveraged to support distributed data storage with no single point of control, controlled access to resources, data integrity, and availability of services.

Blockchain offers a range of built-in features that are well-suited for integrating with our proposed DT-based framework to enhance the security of CPS. For example, the tamper-proof characteristic of blockchain ensures that the data collected and shared within the CPS remains unaltered. The transactions and data operations are recorded on the blockchain and cryptographically secured, protecting against unauthorized modifications, thus providing a transparent and immutable audit trail [16]. Blockchain provides self-executing smart contracts that automate processes within the CPS, such as triggering actions based on sensor data or coded conditions, without a need for intermediaries. Furthermore, blockchain enables a trustless collaboration and co-creation environment within a multi-stakeholder environment using consensus mechanisms to ensure the integrity and authenticity of the transactions. Provenance-enabled blockchain-based DT solutions can support reasoning about the current state of a data object to ensure the traceability and reliability of data, thereby establishing more confidence in the decisions made by the underlying systems [3].

2.3. Threat Intelligence (TI)

In addition to data trustworthiness in DTs for CPS security, context-aware security automation is also essential due to the dynamism of CPS. To this end, intelligence-driven security solutions such as TI can be integrated with blockchain-based DT. Commonly perceived as evidence-based knowledge [17], TI integrates data from multiple sources, such as security information and event management (SIEM) systems, network and system logs, or adversary knowledge bases, to identify threats

and provide actionable security measures for a CPS [18]. By aggregating and analyzing data from these sources, TI can aid in predicting potential security threats and provide a holistic approach to identify and assess vulnerabilities and attack vectors [18]. Thus, combining blockchain, DTs, and TI can establish the triad of trusted and intelligent CPSs with security automation where blockchain acts as a trustworthy data dissemination platform to both physical and twin environments, DT simulates or replicates the states of the physical environment to figure out inconsistencies, and TI extracts actionable insights for situational-awareness and security in CPS (as Fig. 1 shows).

2.4. Building process knowledge for DTs

To fully harness the features of DTs, accurate knowledge about the underlying system and/or process is essential. To achieve accurate knowledge of the physical system, DTs must exhibit sufficient information, i.e., system configurations, data flow and relationship among components, and so on [19]. Such information acts as process knowledge and will determine the level of fidelity of the underlying system. Fidelity can be defined as the degree of resemblance between a physical asset and its twin counterpart in terms of asset data, state, behavior, frequency of updates, and level of abstraction [20]. DT building blocks such as physical assets (that can be devices or processes), real-time sensor-based and historical data, DT services, and semantic technologies [4] together constitute process knowledge of DTs. To this end, the concept of building process knowledge of security-aware DTs can be seen from two perspectives. The first is utilizing a system specification-based model where engineering artifacts of the underlying system are leveraged [8,21–23]. The second perspective is utilizing learning-based or data-driven models [24,25], that leverage learning-based approaches to model the underlying system through both available and synthetic data. Both solutions, i.e., specification-based or learning-based, lack either real-time breach detection or the potential ability to model the correct behavior of physical counterparts. Another approach to building process knowledge is to use IaC. IaC is a process of automating the deployment of environments and system configurations through code [26]. IaC eliminates environment drift, i.e., maintaining the settings of individual deployment environments, thereby automatically reproducing the configuration of deployment environments without inconsistency. Moreover, IaC exhibits (i) idempotence, where the target environment follows the same configuration as that of the current environment, (ii) transparency, making the environment's state explicit and visible; and (iii) reproducibility, enabling easy (re)building of IT infrastructure components [27].

The aforementioned properties are preferable for modeling the DTs. However, to accurately model DTs, we need comprehensive information about the system. Therefore, rather than relying only on specification-based or learning-based approaches, it is essential to build process knowledge of DTs with the confluence of technologies, particularly when leveraging DTs as a security-enhancing enabler. Such well-established modeling of DT through a hybrid of system specification, learning, and IaC can acquire a more comprehensive virtual replica of the system in terms of granularity and accuracy. Furthermore, flexibility in DT modeling is more desirable for realizing the concept of scalable DTs with the functionality to simulate the DT instances based on new use cases in the physical environment. Moreover, IaC also helps automate the security (re)configuration, thereby testing new use cases for security vulnerabilities and threats through TI.

2.5. Summary of related works and research gaps

To date, there exist many works on how: (a) DTs can be used as a security-enhancing enabler covering various use cases, including instruction detection, cyber deception, security risk assessment, digital forensics, testing and training [2,5,13,28], and (b) blockchain-based DTs can ensure data integrity, traceability, availability, and multiple

Table 1

Summary* of existing works addressing: Challenge#1 (Section 3.2), Challenge#2 (Section 3.3), and Challenge#3 (Section 3.4).

Research work	Concept/ Proof of Concept (PoC): Use case	Tool	Essential/Desirable features for modeling DT							Contribution
			Challenge#1		Challenge#2			Challenge#3		
			Data fusion	Trusted sources	Data storage	DT modeling	Trace ability	Data integrity	Security automation	
[8]	PoC: Conveyor belt-driven by motor	CPS Twinning	○	○	○	●	○	○	○	Monitoring of security and safety rules, Generate DT from specification
[21]	PoC: Candy production line	CPS Twinning	○	○	○	●	○	○	○	Intrusion detection, State-based replication
[23]	PoC: Filling plant	MiniCPS	○	○	○	◐	○	○	◐	Incident analysis, Process-based security framework, Formal requirements
[28]	PoC: Mixing tanks	PLCSim Advanced	◐	○	○	◐	○	○	○	Attack detection and localization, Mathematical modeling
[31]	Concept: Bottling plant	n/a	◐	●	●	○	●	●	○	Secure DT data sharing
[22]	PoC: Automotive assembly line	MiniCPS	○	○	●	●	●	●	◐	Anomaly detection, Formal verification
[24]	PoC: Robotic arm	VAR and seq2seq	○	○	○	○	○	○	●	Discuss DT role in intrusion detection
[25]	PoC: Critical infrastructure testbed dataset	PyTorch	○	○	○	◐	○	○	●	Detect anomalies, Generate DT from timed automaton
TRIPLE	PoC: Automotive assembly line	Azure DT	●	●	●	●	●	●	●	Intrusion detection, Securing CPS-DT ecosystem, Formal analysis

Legend: ● Covered; ● Partially covered; ○ Not covered; n/a not applicable.

Note: Selected works focus on (a) focus on industrial use cases, and (b) use DT as an intrusion detection solution with or without blockchain.

stakeholders' collaboration and co-creation activities subjected to access control in the industrial domain [3,29,30]. In this study, from a sample of numerous papers on security-enhancing DTs and blockchain-based DTs, we considered only those focusing on industrial use cases where DT is used as an intrusion detection solution, with or without blockchain Table 1 summarizes the existing works.

Our work builds upon previous studies, such as incorporating rules as defined in [8,22], integrating blockchain-based DTs for ensuring integrity and traceability as per [22,31], considering DT modeling parameters as outlined in [8,21,22], supporting security automation as described in [24,25] and framework verification as in [23]. However, we find that the existing works, focusing on industrial use cases to leverage DT as a security-enhancing enabler (intrusion detection), may not fully address all identified challenges and lack the following essential and/or desirable features for modeling trustworthy DTs:

a. Integration and interoperability of data:

Justification: Effective integration and interoperability of data from multi-modal and heterogeneous sensors are essential prerequisites for realizing a more consistent, comprehensive, and accurate representation of physical counterparts [32] (see Challenge#1: Section 3.2 for details).

b. Trustworthiness of data sources:

Justification: Ensuring the trustworthiness of data sources is critical to ensure the quality of input data to DTs, the integrity of data storage necessary for modeling trusted DTs, the ways for modeling comprehensive DT instance(s), and data traceability and integrity (see Challenge#2: Section 3.3 for details).

c. Automated threat detection by combining DTs and TI:

Justification: Security automation is necessary for timely prediction of cyber-attacks, particularly for critical applications and/or infrastructures due to the dynamic, ever-changing threat landscape (see Challenge#3: Section 3.4 for details).

In contrast to existing works, summarized in Table 1, the TRIPLE framework addresses all essential and/or desirable characteristics identified through Challenge#1, Challenge#2, and Challenge#3. Overall, the TRIPLE framework adds value by (a) securing critical data from both DT and CPS (see Section 3.3), (b) defining comprehensive modeling of DTs (see Section 3.3.1), (c) suggesting combining DT with TI for proactive and reactive threat hunting (see Section 3.4), and (d) formal verification to validate the correctness of the framework (see Section 5).

3. Design of the TRIPLE framework

This section provides an overview of the TRIPLE framework (Section 3.1). It then discusses the challenges (Challenge#1, Challenge#2, and Challenge#3) related to securing CPSs, highlights the significance of the TRIPLE framework in addressing these challenges, and details the essential design requirements of the framework (Sections 3.2, 3.3, and 3.4).

3.1. Overview of the TRIPLE framework

The TRIPLE framework presented in Fig. 2 comprises five key components, including (i) data integration and interoperability, (ii)

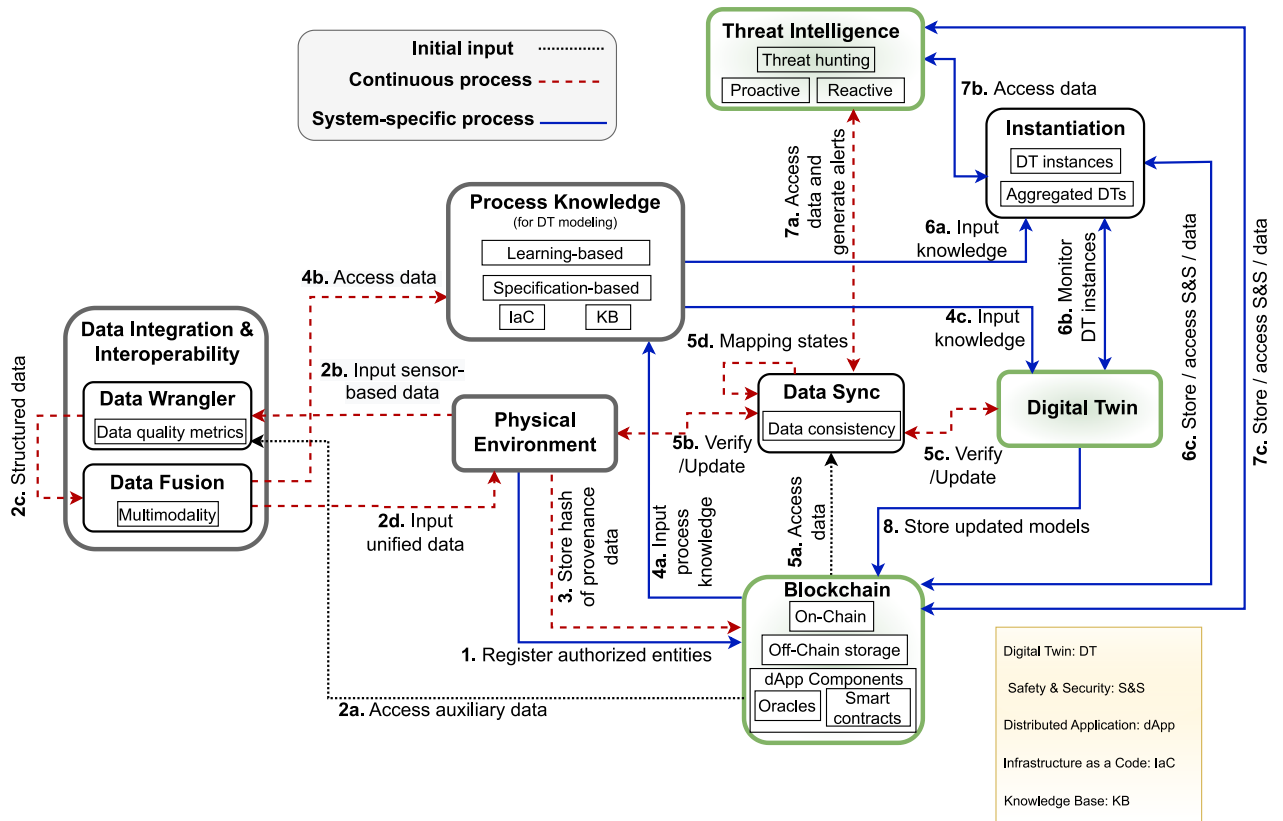


Fig. 2. Overview of the TRusted and Intelligent cyber-PhysicalL systemEm (TRIPLE) framework.

physical asset and its replica, (iii) blockchain (on- and off-chain), (iv) TI, and (v) process knowledge for DT modeling.

The *data integration and interoperability* component is responsible for cleaning invalid, duplicate, or missing data, converting heterogeneous data formats into a unified data structure. This component is also responsible for aggregating data to generate a consistent interpretation of a certain object before inputting data into the physical space. The *physical asset* and its *digital replica* must remain continuously connected and synchronized to ensure ongoing digital-physical mapping. This involves aligning predefined system performance parameters, which are retrieved from storage with real-time sensor data from the manufacturing unit, thereby verifying data consistency in near real-time [33]. The *blockchain ledger* ensures secure data management by storing information immutably and enabling the reliable recall of events. TI identifies vulnerabilities, threats, and potential attack vectors to minimize the threat landscape. Furthermore, the intelligence from the TI module can be further utilized for threat hunting. Note that threat hunting is an iterative process of searching through infrastructures to discover propagation areas of adversarial actions [34]. To this end, *process knowledge* needed to model DTs is drawn from learning-based data, system specifications, and IaC.

The proposed modular TRIPLE framework (Fig. 2) can be adapted depending on the application-specific requirements. Based on the system-specific processes, the data flow can be categorized as (i) the *initial input data* (shown as black dotted lines) that represents one-time input data to the respective entities before initiating the production process. For example, step 5a is performed to access data from blockchain before initiating the production process or between digital-physical mapping gaps; (ii) the *continuous process* (shown as red dashed lines), represents processes that occur continuously, i.e., periodically or depending on defined application-specific requirements during the production process. For example, step 5d of mapping data and states is performed continuously; and (iii) the *system-specific process* (shown as

blue solid lines) represents processes that are not frequent and occur only if there are updates at the end of the production process. For example, step 8 is performed to store DT models in case of updates to the current models. Other steps include instantiating and monitoring DT instances.

The interaction among the components illustrated in Fig. 2 can be summarized as follows.

Step 1 (Trustworthy data sources): The participating entities such as sensors, machines, and humans register as authorized entities on the blockchain [35]. The registration information comprises unique IDs assigned to the entities and their engineering data [8] such as configuration details, threshold data, topology artifacts, etc.

Step 2 (Trustworthy data, data integration, and interoperability settings for CPS operation): To validate data quality, metrics such as threshold checks can be accessed from blockchain (step 2a). The multi-modal and heterogeneous sensor-based data from the physical asset (step 2b) is cleaned and converted to a structured format at data wrangler (step 2c). To reduce reliance on any particular sensor, data from multiple sensors with overlapping fields of view are compared to establish the trustworthiness of data from each sensor. The sensory data is aggregated by data fusion and is inputted to the physical asset to support a consistent and comprehensive representation of the data (step 2d). This step establishes the setup and configuration for the data sync module in the later stages.

Step 3 (Data traceability and reliability): To facilitate track and trace solutions, the hash of provenance data is recorded on the blockchain. Provenance is metadata that records a complete lineage of device data and state at a particular instance of time [36].

Step 4 (Modeling of DTs): To model DTs (step 4c), the process knowledge in the form of the knowledge base (KB) and IaC is acquired from the ledger (step 4a). Additionally, the unified sensor-based data, required by machine learning (ML) models, is provided by the data fusion module (step 4b).

Without loss of generality, we focus on two security operation modes of DTs, i.e., replication and simulation [21].

Step 5 (DT replication mode): In the replication mode, the physical asset (process or device) (*step 5b*) and its replica (*step 5c*) are continuously connected and periodically synchronized (*step 5d*). The purpose of this operation is to track data inconsistencies during the physical process. To create replication mode, a time-dependent digital-physical mapping is performed through the data sync method *step 5d*, where the data consistency method ensures consistent information between the twin and the corresponding physical asset. During the synchronization process, the availability and integrity of data are guaranteed by integrating blockchain (*step 5a*).

Step 6 (DT simulation mode): In the simulation mode, the state of a physical object is observed or tested independently of the physical asset's current state. To model new instances of DTs, process knowledge is acquired from blockchain (*step 6a*). Upon achieving the target configurations determined by the system requirements, the twin is updated (*step 6b*) and eventually the physical asset as well. The generated or existing rules defined based on the historical data and simulations by the plant operators or analysts are stored and accessed from blockchain (*step 6c*). To create simulation mode, an instantiation module is used where DT instances can be customized and executed locally (isolated from the live system), thereby providing flexibility in deployment to simulate unseen use cases. To prevent system failure or degradation caused by security breaches or suspicious data flows, it is crucial to determine the future state of a physical object. This allows for necessary and timely actions, such as scheduling maintenance or switching off compromised devices.

Step 7 (Combining DT modes with TI): To effectively use real-time data streams in DTs, TI accesses data from data sync (through replication mode *step 7a*) and DT instances (through simulation mode (*step 7b*)). ML-based solutions can preemptively detect failures and trigger the appropriate scheduling services in the physical space or model calibration services in the virtual space. These services then carry out necessary measures, such as configuring machine settings based on tool wear data or tuning model parameters to simulate the physical counterpart with high fidelity. The updated models upon calibration and safety and security (S&S) rules must be stored and retrieved from the blockchain before initiating the next production process (*step 7c*). This process ensures the reliability of S&S rules and supports the integration of blockchain with DTs.

Step 8 (Data integrity): The hash of updated models upon calibration are stored on the blockchain (*step 8*).

In the following, we discuss how different components of the TRIPLE framework address the challenges of securing the CPS-DT ecosystem.

3.2. Data integration and interoperability

While integrity and immutability are inherited from the blockchain, ensuring the trustworthiness of data-generating sources is equally important for critical infrastructures [3]. Since this data feeds into other critical sources—such as DT, TI, and process knowledge—for decision-making, neglecting such measures may result in false data entering the system, leading to a garbage-in, garbage-out (GIGO) issue. [37]. Similarly, the inability of the system to conform to data quality metrics (such as data completeness, data precision, data freshness, and so on) can also affect the quality and accuracy of DT, thereby leading to inaccurate decisions. These issues raise the following challenge(s).

Challenge #1: How to (a) ensure the integrity and trustworthiness of data collected from the physical environment, and (b) avoid the insertion of erroneous data into the DT and its physical counterpart?

The following section addresses challenge #1 where we mainly emphasize the trustworthiness of data. To address the above challenge, TRIPLE incorporates a data integration and interoperability module.

The integration and interoperability of data from multi-modal and heterogeneous sensors are significant prerequisites of DT realization [14, 38, 39]. It provides a more consistent, comprehensive, and accurate representation than the single perspective of data [14, 39].

Data integration and interoperability begin by inputting sensor-based data from physical assets to the data wrangler. In TRIPLE, data wrangler is responsible for pre-processing data, i.e., cleaning invalid, duplicate, or missing data. To this end, artificial intelligence (AI)-enabled data curation helps in improving the data quality by implementing intelligent automated methods to fill in the missing data [40]. Supported by blockchain, cross-validation of device data involves, (i) collecting data only from registered devices, and (ii) cross-validating the device data against data quality metrics such as threshold checks, data freshness, and other sensors with overlapping fields-of-view. In addition to these measures, sensors are calibrated periodically for aging management and fault diagnosis at the initial stages of data collection. Given that multi-modal data—such as varying sampling rates, measurement units, and timestamp synchronization—cannot provide a comprehensive interpretation on its own, data fusion is applied [14]. Therefore, the structured data obtained from the data wrangler method is input to the data fusion method, which transforms and fuses the data into consistent interpretations. Note that the choice of a particular fusion method is out of the scope of this paper. Finally, the fused data is fed into the physical asset to execute subsequent steps, such as physical-digital mapping.

3.3. Discussion on blockchain empowered DTs

The process of creating DTs must consider the operation mode of DT, i.e., simulation or replication, data consistency between physical and digital counterparts, and the required degree of fidelity for the DT instance(s). Furthermore, the underlying physical system(s) and their counterpart DT instance(s) which are usually operated by multiple participating entities raise data trust and integrity issues causing the following challenge(s).

Challenge #2: (a) How can trusted DTs be leveraged to accurately reflect the behavior of the physical environment? (b) How to automate the process of (re)creating DT instances that befit different environments and use cases? (c) How can blockchain support data integrity, traceability, and availability in distributed DTs where multiple stakeholders, who partially trust each other, collaborate and co-create activities?

The following section addresses challenge #2 by discussing (a) DT data sync and instantiation modules, and (b) infrastructure-related blockchain design challenges.

3.3.1. Discussion on DT components

The digital-physical mapping, in the replication mode of DT, is performed at the data sync module, whereas the mapping frequency depends on the application-specific requirements [12]. To register changes between the states of the virtual replica and their physical counterpart, the data consistency method ensures the consistency of information in terms of exchanging cyclic data and status updates. This property must hold for replication mode where state changes in a physical entity must be reflected in all replicated instances of the same physical object as pointed out in [12]. The inability to maintain consistency between physical and digital objects may lead to invalid interpretations. To put this into perspective, the termination of the process in the Industroyer 101 and 104 payload malware modules [7] could have had DT replication mode monitoring states, sending commands, and configurations to find out the reasons for terminating the legitimate process as soon as the changes occurred. Thus, if the DT replication mode had been used, the attack could have been prevented earlier or its impact could have been reduced.

Data consistency mainly depends on fidelity, i.e., how precisely a twin mirrors its physical counterpart [38]. However, replication

accuracy depends on various constraints including budget [11], generalization, contextualization [12], granularity level, and synchronization rate. It is also worth noting that the replication and data consistency in environments that need almost real-time data exchange between the physical system and DT calls for ultra-fast communication networks such as 5G and beyond [41]. Fidelity also plays a pivotal role in the scalability of the CPS. For instance, to replicate 5G and 6G infrastructure at scale, high-fidelity DT will be needed [41]. Furthermore, complex systems such as 5G and 6G infrastructure will benefit from high-fidelity DTs where different use cases of these technologies and their role in different verticals can be realized.

The instantiation module, in essence, supports the simulation mode of DT, which provides *trial and error* approach, i.e., rerunning simulation with user-defined settings to monitor the system's emergent behavior. This module also provides flexibility in deployment to simulate new use cases by running them independently of their physical counterpart. The DT instances can be customized and executed locally (isolated from the live system) because this module can facilitate experimenting with (re)configurations. For instance, these experimentations include, through simulation mode, integrating new devices before connecting them to the physical environment, and testing devices to figure out the reasons for cascading failures and hence localizing the compromised device(s), to name a few. Therefore, the instantiation module can implicitly promote the concept of scalability in DTs. In the instantiation module, the use of IaC can automate the process of recreating instances (as step 6a of Fig. 2 shows) and enhancing system testing strategies through multi-fidelity DTs [42].

Multiple instances of DTs, where each instance mirrors a physical asset, can be aggregated to realize the scale-up version of the underlying process and/or environment (e.g., deployment of a 6G network in a metropolitan city). To enforce a security-by-design approach in terms of data integrity and reliability, the input data (such as predefined simulation parameters or rules) required to instantiate use cases are accessed from the blockchain. The newly generated or modified S&S rules are stored in the blockchain. To ensure data traceability and access privileges, the information about the accountable entity generating or updating rules for the respective physical objects is also maintained in the blockchain, as suggested in [22]. For instance, the Industroyer [7] could be detected during the reconnaissance phase, i.e., the installation of the main backdoor and additional backdoor malware components if proper checks are exercised and access data is traceable.

3.3.2. Discussion on blockchain-based integration

Due to the key benefits of blockchain, including integrity, immutability, traceability, transparency, automation through smart contracts, and multiparty access control, blockchain can transform DTs to enable trustworthy data dissemination and data management [29]. The TRIPLE framework mainly leverages blockchain to ensure the integrity and traceability of data inputted to the physical environment and DTs in the form of rules, models, provenance data, and auxiliary data. Furthermore, unlike traditional centralized solutions, blockchain is resilient against a single point of attack or failure and thus can tolerate and recover accidental or malicious failures without causing data or service unavailability [16]. Thus, blockchain-based solutions are a viable option for DTs that are operated and maintained in a multi-stakeholder environment where centralizing trust in a traditional database is a drawback. In other words, blockchain provides consistency of data across partially trusted participants.

Because data is accessed by the physical environment and DT throughout the physical process, it is essential to analyze blockchain-related design challenges. To gain practical insights into CPSs, blockchain-related design challenges may cover the deployment of blockchains, blockchain types, consensus mechanisms, scalability, storage, and other performance constraints. The deployment of blockchain may vary based on the underlying architecture. Considering resource-constrained IoT devices, deploying blockchain on fog or edge nodes

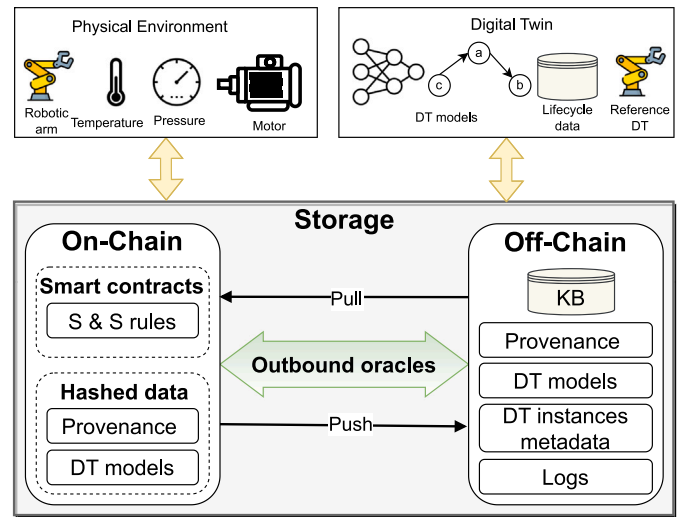


Fig. 3. High-level illustration of on-chain and off-chain storage with respective data components from both physical and twined environments.

[43] to carry out resource-intensive tasks (such as running a consensus algorithm or executing smart contracts) can provide a trade-off between trust centralization and overhead [16]. For instance, before initiating the production process, the relevant data can be fetched from the storage (as step 5a of Fig. 2 shows). Additionally, we can limit the frequent time-consuming access to the blockchain-based storage by explicitly separating the real-time dynamic (or behavioral) data and the less dynamic (or static) data [37]. This strategy may raise the question of leveraging blockchain as mere data storage for DTs. However, this strategy is adopted because the real-time data evolves frequently with each lifecycle phase and usually amasses in intervals, such as deployed sensors and actuators data providing the asset's actual state, while static data changes infrequently with time along the lifecycle of the real-world counterpart, such as provenance data, system historical data, device configuration settings, access levels, and policies.

Fig. 3 presents the details of the storage components in the TRIPLE framework that integrate the on-chain and off-chain storage concepts. On-chain storage refers to storing data on the blockchain and hence becomes immutable. However, blockchains have limited storage capacity and become inefficient if large amounts of data (i.e., DT models, DT instances metadata, logs) are stored on-chain. Also, such data can increase transaction fees and maintenance costs [44]. To address these challenges, we use off-chain storage to store large data outside the blockchain, e.g., in an external database. In our case, even though the data is stored off-chain, a cryptographic hash of the data is stored on-chain. This hash acts as a secure reference to verify that the off-chain data has not been tampered with [45]. To enable interactions between on-chain and off-chain, we use blockchain oracles. Blockchain oracles are classified into four types, determined by the direction of data flow (inbound or outbound) and the method of initiation (push or pull) [46]. In this work, we integrate the pull- and push-based outbound oracles. For instance, pull-based outbound oracles request data from the blockchain (on-chain) and push-based outbound oracles deliver the data to the external system (off-chain).

In addition to application- or user-specific requirements, critical data and sources that can facilitate the track and trace solutions for data-driven CPS must be considered. For instance, TRIPLE primarily stores the hash of provenance data, models, and S&S rules as on-chain storage (as Fig. 3 shows). The selection of apropos blockchain further depends on scalable solutions that become critical due to the growing number of actors and activities in the CPS. To enable scalable solutions, on-chain approaches based on blockchain data structures—such as

Directed Acyclic Graph (DAG)-structured blockchains—are preferable to traditional chain-structured blockchains. [47]. In terms of implementation, most DAG-structured blockchains may not yet be mature enough for mass adoption. For instance, IOTA is still facing centralization issues [47]. Distributed hash table (DHT) is another choice for distributed data storage, where data is spread across multiple nodes in a network and can be efficiently retrieved using a key–value pair system. However, DHT lacks inherent security and immutability and does not provide strong consensus mechanisms or fault tolerance, making it vulnerable to routing and storage attacks, data tampering, data retrieval inconsistencies, and potential malicious attacks [48,49]. In contrast, chain-structured blockchains, e.g., Hyperledger or Ethereum, incorporate robust cryptography techniques, maintain decentralized and distributed networks, provide data immutability, and integrate strong consensus mechanisms that ensure network participants agree on the ledger's true state [45]. These inherent features of blockchain make it a suitable choice for our case where data integrity, security, and trust are critical. However, most chain-structured blockchains require the integration of off-chain storage to tackle scalability and performance challenges (as Fig. 3 shows), and they have not yet achieved quantum-safe security [50].

To sum up, considering the infrastructure-related blockchain design challenges [47], there exists no comprehensive solution that efficiently supports the diverse requirements of CPSs, including throughput, scalability, storage efficiency, and security [16]. Therefore, the design and operation requirements of blockchain-based DT in an underlying use case are subjected to thorough investigation. Furthermore, a trade-off among factors including (a) performance (communication, processing, and storage overheads), (b) maintainability (infrastructure and resource management, and scalability), (c) safeguarding lifecycle data (security, privacy, and reliability), and (d) DT services is needed.

3.4. Combining DTs with TI

While DT security operation modes support various security-enhancing use cases [2], they pose limitations. For instance, the simulation mode relies on the user-specified assumption regarding simulation parameters, whereas the replication mode is constrained by state/time-dependent synchronization [12]. Consequently, creating future scenarios, particularly without direct observation data, is challenging for DTs [40]. The timely prediction of cyberattacks is especially crucial for critical infrastructures and becomes difficult due to the dynamic, ever-changing threat landscape. Moreover, security automation in connection with contextual information is also desirable. To this end, the issues mentioned above raise the following challenge(s).

Challenge #3: How to achieve (a) security through DTs and (b) security of DTs simultaneously?

The following section addresses challenge #3 (a) by combining DT replication and simulation mode with TI and blockchain integration to address challenge #3 (b). TI integrates data from multiple internal sources, such as SIEM systems, network traffic analysis, and user behavior analytics, as well as external sources, including TI feeds, open-source intelligence, and dark web monitoring, to analyze threats and provide actionable insights [18]. Threat hunting utilizes TI as a KB [51] to formulate attack hypotheses and actively search for potential threats in a CPS environment. The TRIPLE framework recommends leveraging TI alongside the replication and simulation modes of DTs. For example, during the replication mode, the time-dependent data sync module may cause data inconsistencies between physical and digital objects. These delays could be due to various reasons, including network latency or computational and processing constraints. Therefore, threats posed during this phase are covered by reactive threat hunting. In contrast, the TI module performs real-time monitoring of threats, which can be covered by proactive threat hunting to provide timely indications of presumably ongoing attacks. Based on the collected data points,

the security operation center (SOC) team can assess the risk of security events and make informed triage decisions. Furthermore, threat hunting can provide valuable insights into new attack vectors and investigation outcomes in the form of indicators of compromise (IoCs) or tactics, techniques, and procedures (TTPs) to the security community. This includes contributions to frameworks such as adversarial tactics, techniques, and common knowledge (MITRE ATT&CK), as suggested by [52].

How DTs can support TI? One potential use case of DT replication mode with TI is the creation and maintenance of a replica of supervisory control and data acquisition (SCADA) systems to support security. This replication can encompass the behavioral aspects and operational environment of the SCADA systems. In this regard, the data sync module (see Fig. 2) can determine inconsistencies between a physical process or device and its digital counterpart. For instance, in the case of Industroyer malware, the multi-stage attack components such as backdoors and payloads can be detected in the initial phases. With real-time monitoring, such inconsistencies can be triggered by reactive threat hunting (as step 7a of Fig. 2 shows). Disjoined from the physical system, devices, or processes can also be analyzed in the instantiation module and integrated into the DT (as step 6b of Fig. 2 shows). As observed in the Industroyer, a destructive data wiper module wipes out ICS configuration files, thereby rendering the system unbootable [7]. With twinned SCADA systems, even if the inconsistencies at the data sync module go unnoticed, proactive threat hunting provides timely indications of presumably ongoing attacks. In the worst case, attack repercussions can be reduced as the SCADA system can be restored from DT archived versions. Moreover, it can also reproduce attack scenarios for further investigation.

Similarly, the DTs simulation mode along with TI can serve as a cascade testing platform through which individual processes or devices can be tested, for instance, simulating a circuit breaker of a power grid substation to monitor and analyze its state transitions, e.g., open, close, half-open, under different configurations for use cases such as energizing or de-energizing substation. Moreover, multiple DT instances, such as circuit breakers, protective relays, remote terminal units (RTUs), intelligent electronic devices (IEDs), and others can be aggregated to study the overall system and analyze its overall lifecycle. The required instances (subsystems) can be constructed by using the process knowledge (step 6a in Fig. 2). DT instances can also provide threat information to the TI module, enabling it to analyze patterns and develop security rules that detect and respond to anomalies, such as abnormal state transitions in critical components. For example, in a power grid substation, if a DT instance of a circuit breaker detects abnormal state transitions, such as unexpected fluctuations between open and closed states during a simulation, this data can be relayed to the TI module. The TI module can then analyze these anomalies to develop new security rules, ensuring that similar threats, such as potential tampering or malfunctioning devices, are swiftly identified and mitigated in the real grid.

DTs and their physical counterparts are characterized by bidirectional data flow [33], therefore it is essential to ensure the trustworthiness of both data sources. Given this fact, the TI module must be able to integrate and analyze data from both physical and twin environments. In this regard, in the proposed TRIPLE framework, the TI module accesses data from the data sync method that mirrors the sync states of both physical and twin objects and can analyze both spaces' behavior. Furthermore, this strategy can achieve both objectives, i.e., 'security through DTs' and 'security of DTs'.

Another interesting aspect could be to sort out the problem of false alarms or alert fatigue without completely ignoring or overloading security alerts. Note that the capability of DTs to reproduce experiments may help in detecting false alarms, however, mitigating such problems requires consideration of other solutions, such as aggregation of anomalies [52].

Table 2
Summary of notations used in the proof of concept (Section 4).

Notation	Explanation	Notation	Explanation
$\tau_{\{min,max\}}$	Threshold data	D_{prov}	Provenance data
A_{ID}	Asset ID	$(S/M)_{ID}$	Sensor/Machine ID
off_{chain}	Off-chain storage	on_{chain}	On-chain storage
SS	S&S rules	D_{A_i}	Timestamped asset data
		DT_i^{model}	DT models of respective assets
$Instance$	DT instance	DT_{MODEL}	Reference DT model (aggregation of DT_i^{model})
D_q	Structured data	$D_{S,S}$	Data syntax and semantics
$\chi_{configs}$	Asset configurations	χ_{specs}	User/application specifications
$f\omega$	Wrangler function	$f_{s_i}^{op}$	Three-step function with operations: ingest ($f_{s_i}^{ingest}$), transform ($f_{s_i}^{transform}$), and feed ($f_{s_i}^{feed}$)
		$S_{contract}$	Smart contract
DT_{PK}	Process knowledge	$\mathcal{KB}$	Knowledge base
Ont	Ontology		

We anticipate that fully realizing and implementing the TRIPLE framework may exceed the scope of this paper; however, at this stage, it is imperative to conduct a simplified proof of concept (Section 4), based on the use case (Section 4.1) and formal analysis (Section 5), to validate the correctness of the functional design and ensure it meets the key objectives.

4. Proof of concept

In this section, we present a preliminary prototypical implementation of the TRIPLE framework. First, we describe the high-level architecture of the use case (Section 4.1). Then, with reasonable assumptions, we discuss the steps for the creation and deployment of DT (Section 4.2) and integration of blockchain (Section 4.3) to ensure the integrity of critical data. Finally, we explain how to use DT for visualization of services and querying of data (Section 4.4). Fig. 4 presents the steps required to develop a working prototype, including the authentication settings to connect with Azure IoT Hub, temperature sensor configurations, robotic arm telemetry, integration of blockchain, and DT services. Table 2 summarizes the notations used in this section.

The implementation of DTs depends on three factors: (a) the objective of leveraging DTs in the underlying use case, (b) available resources and technologies, and (c) the pre- or post-existence of DTs in comparison to physical systems. For the TRIPLE framework, the objective is intrusion detection in the automotive assembly line (addressing factor a). We choose Microsoft Azure Digital Twins (ADT)² as a DT platform that models DTs in a cloud computing environment following platform-as-a-service (PaaS) architecture and enables physical twin monitoring in a simulated environment. For the management of DT's instances, the ADT service offers control and data application programming interfaces (APIs) [53]. Control APIs handle tasks like adding or removing DT instances and endpoints, while data APIs are ADT APIs for managing models, twins, graph queries, and event routes (addressing factor b). We consider DT post-existence to help in the operational phases of the physical system (addressing factor c). The source code of the prototype is available on GitHub.³

4.1. Case study: Automotive assembly line

We consider an assembly line in the automotive industry as a CPS use case (as illustrated in Fig. 5). In this exemplary physical process, a motor-driven conveyor belt system moves the chassis from Station A (loading stage) to Station B (welding stage) for spot welding by a robotic arm. The physical assets have sensors to measure or monitor their operations based on predefined S&S rules (SS). For example, Sensor2 (Object detection) detects the presence of chassis to maintain only a certain number of chassis on the belt, and Sensor1 (Velocity) monitors the belt's velocity against a specified motor speed. Similarly,

Sensor4 (Pressure) and Sensor5 (Temperature) measure the pressure and temperature during welding and are bound by a threshold to avoid material deterioration. SS rules cover data consistency, integrity, and threshold checks. The PLCs (PLC1, PLC2) and human-machine interfaces (HMIs) (HMI1, HMI2) are used to command and control the physical assets. A DT of the automotive assembly line use case is essential for several reasons. For instance, the twinned environment can detect inconsistencies in physical assets and their operations. It can also be employed to test the asset's safe states against integrity or consistency checks.

4.2. DT design and deployment

In the following, we describe a step-by-step working prototype to demonstrate how physical assets, i.e., robotic arm and temperature sensor in our use case can be modeled in a twinned environment.

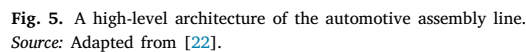
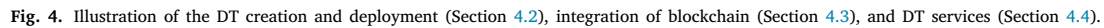
Algorithm 1: Algorithm 1 outlines the steps involved in designing (phase I), loading data into (phase II), configuring (phase III), storing (phase IV), and deploying (phase V) the DT model.

Phase I: Step 4 in Fig. 4: First, for each asset ($Asset_i$), define a schema. A schema refers to the structured representation that defines the asset's attributes, properties, relationships, and behaviors within the DT framework. Second, construct ontology (Ont) from the schema of the respective asset. An ontology defines the formal representation of a domain's concepts, categories, properties, and relationships. For instance, an ontology of the robotic arm temperature sensor is illustrated in Fig. 6. Note that the ontology can be constructed from a knowledge base ($\mathcal{KB}$) and process knowledge (DT_{PK}). Third, cross-check Ont against data syntax and semantics ($D_{S,S}$). We utilize the Digital Twin Definition Language (DTDL) validator to check the model document's correctness and ensure the defined schema is valid. Finally, the DT models (DT_i^{model}) of respective assets, i.e., robotic arm and temperature sensor are created (phase I: step 1d). We created a DT model of two welding robotic arms, each with a temperature sensor (Sensor5). These individual DT models (DT_i^{model}) are then aggregated to generate DT_{MODEL} . The purpose of creating individual models of each asset is to allow granular representation and customization as needed in different DT instances.

With ADT, plant operators can design their twin graph represented by the DTDL. Table 3 displays the fields in the DT modeling of the robotic arm, where parameters prefixed with @ denote required fields. DTDL is based on JavaScript Object Notation for Linked Data (JSON-LD) schema for describing twined IoT devices. For example, in our use case, we have two models (RoboticArmInterface.json, TemperatureSensor.json) that have properties (e.g., ID, RoboticArmLocation, Temperature, TemperatureAlert, DeviceState) (see Fig. 6), and relationship (e.g., rel_has_temperature_sensors), thereby indicating the robotic arm has a temperature sensor. Note that we present a minimal setup of the use case depicted in Fig. 5; however, the other assets, e.g., sensors (velocity, object detection, and pressure), and devices (conveyor belt and chassis) can be virtualized similarly.

² <https://docs.microsoft.com/en-us/azure/digital-twins/overview>

³ <https://github.com/mubashar-iqbal/triple-digital-twin>



manage. First, the IoT Hub requires the registration and authentication of IoT devices. This is achieved using a symmetric key attestation technique supported by Microsoft ADT, which enables device identification through shared access tokens provided by IoT Hub [54]. The symmetric key is used to generate HMAC-SHA256 signature strings, employing the SHA-256 cryptographic hash function for shared access tokens [54]. Detailed steps for establishing a connection between an IoT Hub and registered IoT devices can be found in [Appendix A: Listings 3 and 4](#). Next, employ the Azure function (referred to as a three-step function ($f_{s_i}^{op}$) to ingest, transform, and feed data into the ADT (phase II: steps 8a to 8c). In the first step ($f_{s_i}^{ingest}$), the data is received from Azure IoT

Table 3
Details of the robotic arm DT model illustrating the fields of ontology shown in Fig. 6.

Field	Description
@id	Unique identifier of the model. We use the available format (<i>dtmi</i> : < domain > : < unique - model - identifier > ; < model - version - number >) to define the model @id.
@type	Identify the type of information being described in the model. We use the Interface type for both models.
@context	Defines the namespace and context of the JSON-LD document. Every model should define the context (dtmi:dtdl:context;2).
displayName	Define a user-friendly name for the model.
contents	Used to place the model data.

```

"@context": "dtmi:dtdl:context;2",
"@id": "dtmi:robot:digitalroboticarm:temperature_sensor;1",
"@type": "Interface",
"displayName": "Temperature Sensor Interface Model",
"contents": [
  {
    "@type": "Property",
    "name": "Id",
    "schema": "string",
    "description": "Sensor Id",
    "writable": true
  },
  {
    "@type": "Property",
    "name": "Temperature",
    "schema": "double",
    "description": "Current measured temperature",
    "writable": true
  },
  {
    "@type": "Property",
    "name": "TemperatureAlert",
    "schema": "string",
    "description": "Outside of desired temperature alert",
    "writable": true
  },
  {
    "@type": "Property",
    "name": "DeviceState",
    "schema": "string",
    "description": "Functional and non-functional state of robotic arm",
    "writable": true
  }
]

```

Fig. 6. Ontology of a robotic arm temperature sensor model defined in JSON-LD: Excerpt from Microsoft ADT.

Hub. In the second step ($f_{s2}^{transform}$), the data is transformed into an ADT readable format using JSON document patch format. In the third step (f_{s3}^{feed}), the transformed data into the DT models.

Phase III: To configure DT_i^{model} , specify process knowledge (DT_{PK}) which may include user or application-specific data (χ_{specs}), system/asset configurations ($\chi_{configs}$), or learning models. Then define SS rules that can be categorized as (i) data, (ii) operational, and (iii) functional (as Table 4 shows). Given that CPS comprises a system, network, and application, rules can be further categorized accordingly. However, we will be focusing more on data rules at the system and application level. To construct provenance (D_{prov}), metadata field ($meta\_data$) can be used. For instance, Fig. 7 presents the timestamped $meta\_data$ and $\chi_{configs}$, including sensor ID (id of the sensor), temperature (value), temperature alert (low), and device state (non-functional). The $\chi_{configs}$ are depicted in Fig. 7 as twin properties that enable DT_i^{model} to adapt to the real-time values received from the sensors.

Phase IV: Steps 6 and 7 in Fig. 4: To ensure the data integrity of the DT and CPS-related data, store hashes of DT_{MODEL} , SS and D_{prov}

following the steps in Phase II: Algorithm 2. To store DT_{PK} , D_{A_i} , and $\mathcal{KB}$ following the steps in Phase III: Algorithm 2 (see steps discussed in Section 4.3: Algorithm 2).

Phase V: Step 7 in Fig. 4: Deploy the aggregated models (DT_{MODEL}) in the Microsoft ADT Explorer (see steps discussed in Section 4.4: Algorithm 4).

4.3. Integrating blockchain with DT

We create a decentralized application (dApp) on the Ethereum blockchain that assists us in establishing and obtaining S&S rules (e.g., temperature threshold). We selected the Ethereum blockchain due to its mature ecosystem and self-executing smart contracts capabilities. Furthermore, its proof of stake (PoS) consensus mechanism provides transaction integrity, transparency, and resistance to censorship, fraud,

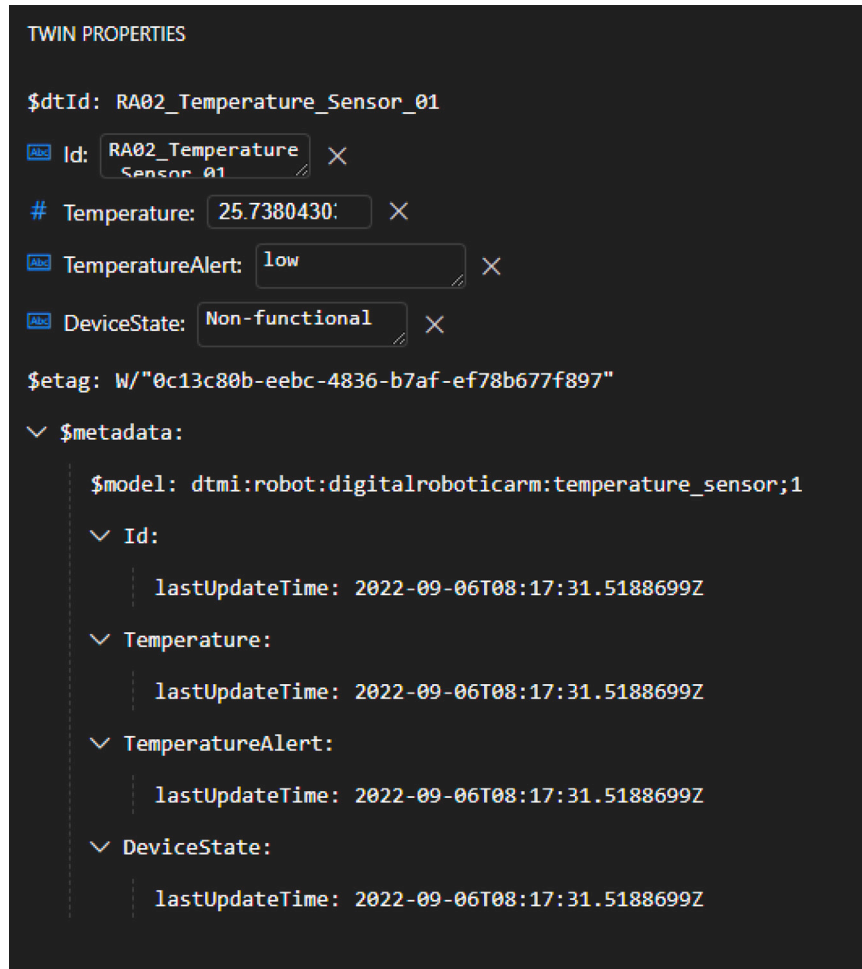


Fig. 7. Excerpt of the DT robotic arms' timestamped metadata from Microsoft ADT.

Table 4
Examples of S&S rules for an automotive assembly line.

S&S rules category	Purpose	Example (Automotive assembly line: Fig. 5)
Data	Establish baselines to identify and ensure benign behavior.	System: Set threshold values for robotic arm welding parameters using Sensor4 (Pressure) and Sensor5 (Temperature). Network: Monitor data packets for anomalies in data transmission between conveyor belt sensors. Application: Validate data integrity from Sensor2 (Object detection) to ensure the correct number of chassis on the conveyor belt.
Operational	Define the functional and non-functional states of assets to ensure proper operation, including access control.	System: Specify acceptable operational ranges and define normal operating conditions for robotic arms. Implement role-based access control (RBAC) to restrict access to critical production control systems. Network: Ensure secure communication protocols between PLCs and HMIs to avoid disruptions. Use firewalls and network segmentation to protect communication channels. Application: Implement criteria for emergency shutdowns in the control software for safety compliance. Limit access to production control software to authorized personnel.
Functional	Ensure the proper functioning of all components within the system.	System: Regular firmware updates and patches for PLCs and HMIs to avoid vulnerabilities. Network: Encrypt data transmitted between sensors, PLCs, and HMIs. Monitor network traffic for unauthorized access and anomalies. Application: Secure system configurations and user authentication mechanisms.

Algorithm 1 Design and deploy a DT model

Input: $Scehma$, Ont , $\mathcal{KB} \supseteq \{\text{historical, contextual, and domain data}\}$,
 $DT_{PK} = \{\chi_{spec}, \chi_{config}, \text{learning models}\}$, D_η (derived from f_ω)
Output: $DT_{MODEL} \supseteq DT_i^{model}$, D_{Prov} , SS

Phase I: Design DT model

- 1: **for** each $Asset_i \supseteq S_{ID} / M_{ID}$ **do**
 - a: Define $Scehma$
 - b: Construct Ont from respective $Scehma$ $\triangleright Ont$ can be derived from $\mathcal{KB}$ and/or DT_{PK} .
 - c: Crosscheck Ont against $D_{S,S}$ $\triangleright D_{S,S}$ include data type, structure, formats, etc.
 - d: Generate DT_i^{model} $\triangleright$ A DT model for respective asset.
- 2: **end for**
- 3: Aggregate DT_i^{model} to generate DT_{MODEL} $\triangleright DT_{MODEL} \supseteq DT_i^{model}$

Phase II: Load data to DT model

- 4: **switch** s **do**
 - 5: **case** IoT_Hub $\triangleright$ TRIPLE case
 - 6: IoT Hub setup $\triangleright$ Register IoT devices and authenticate using shared access tokens.
 - 7: **function** $LOAD\_DT\_DATA\_IOT\_HUB(data)$
 - 8: Obtain timestamped asset data $D_{A_i^t}$ from Azure IoT Hub through $f_{s_i}^{op}$
 - a: $f_{s_1}^{ingest} : D_{telemetry} \triangleright$ Obtain telemetry data.
 - b: $f_{s_2}^{transform} : D_{telemetry} \mapsto D_{A_i^t} \triangleright$ Transform data.
 - c: $f_{s_3}^{feed} : DT_i^{model} \leftarrow D_{A_i^t} \triangleright$ Feed data to DT.
 - 9: **end function**
 - 10: **case** $Assets\_data$
 - 11: **function** $LOAD\_DT\_DATA\_ASSETS(data)$
 - 12: D_η derived from f_ω $\triangleright$ Physical assets data (preferable for replication mode). Justification of *Challenge#1*.
 - 13: **end function**
 - 14: **case** $Dataset$
 - 15: **function** $LOAD\_DT\_DATA\_DATASET(data)$
 - 16: $public\_dataset \triangleright$ Use publically available datasets.
 - 17: **end function**

Phase III: Configure DT model

- 18: Specify DT_{PK}
- 19: Define $SS \leftarrow \{data, operational, functional\} \triangleright$ Rules can be derived from DT_{PK} and $\mathcal{KB}$.
- 20: Define $D_{prov} \supseteq meta\_data \{D_{A_i^t}, DT_{PK}\} \triangleright$ Construct provenance data from metadata to track and trace sources. Justification of *Challenge#2*.

Phase IV: Store DT model

- 21: To store DT_{MODEL} , SS , D_{prov} call Phase II: Algo. 2
- 22: To store DT_{PK} , $D_{A_i^t}$, and $\mathcal{KB}$ call Phase III: Algo. 2

Phase V: Deploy DT model

- 23: Deploy DT_{MODEL} $\triangleright$ Deploy DT model in a simulator.
- 24: Access DT_{MODEL} through Scoreboard (Call Algo. 4)

and third-party interference [45,55]. We utilize Microsoft Azure Storage platform as an off-chain storage to address the scalability and performance issues associated with on-chain storage. We integrate Microsoft Azure Storage because it offers cloud-based storage that seamlessly integrates with ADT and provides HTTPS-based REST APIs for secure and convenient data access. Specifically, off-chain storage is used to obtain the DT models and schema, as well as storing timestamped metadata and process logs, which can be large and complex.

Algorithm 2 Store DT and CPS data on the blockchain

Input: DT_{MODEL} , D_{Prov} , $\mathcal{KB}$, DT_{PK} , SS , $Oracles_{OUT, Pull}$,
 $Oracles_{OUT, Push}$, $S_{contract}$
Output: on_{chain} , $of f_{chain}$

Phase I: Smart contract based S&S rules (on_{chain})

- 1: **modifier** onlyOwner() $\triangleright$ To specify only the contract owner can initiate this request.
- 2: **constructor**
 - a: owner = msg.sender $\triangleright$ Access control to restrict only the owner of the $S_{contract}$ can set SS .
- 3: **define** $value_{min}$, $value_{max}$ $\triangleright$ Temperature threshold values (min and max).
- 4: **function** SETTEMPERATURETHRESHOLD(threshold) onlyOwner
- 5: $value_{min} = threshold.min\_temperature$ $\triangleright$ Set minimum threshold value.
- 6: $value_{max} = threshold.max\_temperature$ $\triangleright$ Set maximum threshold value.
- 7: **end function**
- 8: **function** GETTEMPERATURETHRESHOLD
- 9: **return** $value_{min}$, $value_{max}$ $\triangleright$ Return min. and max. temperature.
- 10: **end function**

Phase II: On-chain storage (on_{chain})

- 11: **For** other lengthy data:
 - a: Compute hash: $h\{D_{Prov}\}$, $h\{DT_{MODEL}\}$, and $h\{SS\}$
 - b: Send $\{h\{D_{Prov}\}, h\{DT_{MODEL}\}, h\{SS\}\} \rightarrow Oracles_{OUT, Push} \triangleright$ Send hash of data through outbound oracle to save it off-chain with metadata. Call Algo. 4.

Phase III: Off-chain storage ($of f_{chain}$)

- 12: Store $\mathcal{KB}$, DT_{PK} , and $D_{A_i^t}$

Phase IV: Provenance-enabled blockchain (traceability)

- 13: Derive $meta\_data \leftarrow \{\tau_{min, max}, DT_{MODEL}, DT_{PK}, D_{A_i^t}\}$ Call Algo. 1

However, we still maintain a reference (e.g., hash ID) of such data on-chain (Algorithm 2-Phases II and III). We deploy the dApp on the Goerli test network, which provides proof of stake (PoS) consensus following the default settings of the Ethereum blockchain (e.g., network size, block time and size, gas limit, difficulty, and block reward). Using the Netherium library, we interact with the dApp and Ethereum nodes. For instance, we fetch the defined S&S rules from the blockchain before initiating the simulation process. Particularly to our use case, blockchain provides a tamper-resistant (immutable) environment for storing and exchanging S&S rules, as well as enables decentralized, transparent, and verifiable transaction processes, making it easier to track and audit S&S rules across multiple industrial stakeholders. Thus, using blockchain ensures S&S rules reliability, ultimately strengthening the rationale for integrating blockchain with DTs. Moreover, S&S rules implemented through smart contracts cover the data integrity checks, access privileges, and triggering the fail-safe state in case of failures, thereby supporting industrial automation.

Fig. 8 represents the interaction between off- and on-chain components, DT instance, dApp, smart contract-based rules, and multiparty stakeholders. The system initiates by registering contract owner nodes on the blockchain (step 1). Subsequently, these contract owner nodes deploy the smart contract (step 2). Once deployed, the contract owner sets the appropriate threshold values (step 3). The ADT, functioning as an off-chain client, then requests these threshold values from the blockchain (step 4). This threshold request is prepared using a Web3 client (step 5), which subsequently sends the request to oracles. The

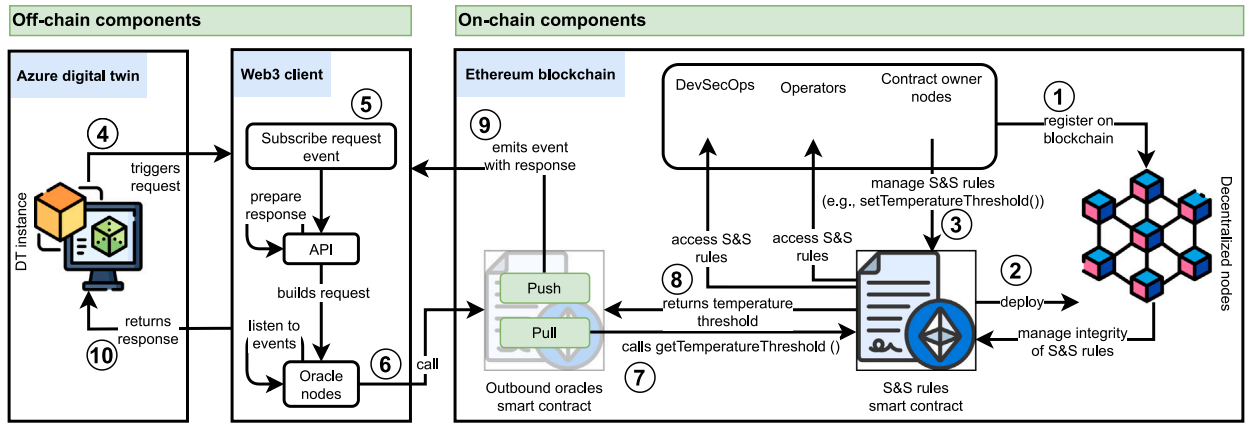


Fig. 8. Integration of blockchain with DT showing an interaction between off-chain and on-chain components, DT instance, dApp, smart contract-based rules, and multiparty stakeholders.

oracle interacts with the smart contract to retrieve the temperature threshold values (step 6). After obtaining the values from the smart contract, the oracles emit these values in response to the subscriber's request event call (steps 7, 8, and 9). Finally, the ADT receives the response containing the threshold values (step 10). The illustration presented in Fig. 8 is discussed in Algorithm 2 and Algorithm 3, which describes the required steps for integrating blockchain-based DTs.

Algorithm 2: Algorithm 2 outlines the steps involved in defining smart contract-based S&S rules (phase I), storing data on-chain and off-chain (phase II and II), and maintaining provenance.

Phase I: To define a smart contract ($S_{contract}$), we use the modifier function in Solidity⁴ that allows extra conditions on certain operations within the deployed smart contract. In our case, we use `msg.sender` to indicate the contract creator (contract owner) who deployed the smart contract and can execute the `setTemperatureThreshold` function. The `setTemperatureThreshold` function sets the minimum and maximum thresholds for allowed and safe temperatures during working mode. The `getTemperatureThreshold` function returns the defined minimum and maximum thresholds. For the `getTemperatureThreshold` function, we did not use the `onlyOwner` access control to allow other stakeholders within the system to view the set values. Note that we have illustrated one example of $S_{contract}$ for setting and retrieving S&S rules; other rules can be defined similarly.

Phase II: In this phase, we obtain data including D_{Prov} , DT_{MODEL} , and SS from Algorithm 1. We compute the hashes of this data and store them on-chain. Then, the system sends a hash of data and metadata through an outbound oracle to save it off-chain. We store metadata off-chain due to its large size, which helps address scalability issues related to performance and the limited storage capacity of blockchains.

Phase III: For other data sources, including knowledge base (KB), process knowledge (DT_{PK}), and timestamped asset data (D_{Ai}) we store them on off-chain storage. Similarly, we store this data off-chain due to its large size and to address scalability issues associated with blockchains.

Phase IV: This phase derives metadata ($meta\_data$) comprising $\tau_{min,max}$, DT_{MODEL} , DT_{PK} , and D_{Ai} . The $meta\_data$ field can be used to construct provenance (D_{prov}), which will be stored on-chain as a hash while maintaining full details off-chain (Algorithm 2-Phase IV). This provenance-enabled technique allows traceability during data acquisition and dissemination.

Algorithm 3: Algorithm 3 outlines the steps involved in data requests from off-chain to retrieve the data from on-chain. We use pull-outbound and push-outbound oracles to enable interactions between Off_{chain} and On_{chain} .

Algorithm 3 Blockchain oracles

Input: Data request from off_{chain}

Output: Data retrieved from on_{chain}

Phase I: Pull-Outbound

- 1: modifier `onlyOwner()` $\triangleright$ To specify only the contract owner can initiate this request.
- 2: constructor
 - a: `owner = msg.sender` $\triangleright$ Access control to restrict only the $S_{contract}$ owner can initiate a data request to on_{chain} state from the deployed $S_{contract}$ once the data request is triggered from off_{chain} state.
- 3: define `response` $\leftarrow$ ""
- 4: `OraclesOUT,Pull` $\triangleright$ Register pull-outbound event of oracle.
 - a: `response` $\leftarrow$ `requestProcessing(authenticityProof)` `onlyOwner` $\triangleright$ Identify API endpoint, endpoint authentication, and checking the integrity of request.
 - b: Register $\leftarrow$ `Request_Eventblockchain` $\triangleright$ Request event (register on blockchain).

Phase II: Push-Outbound

- 5: Data verification (response) $\triangleright$ Validating digital signatures, checking for tampering or manipulation, and ensuring data consistency.
- 6: Prepare data (response) $\triangleright$ Data formatting.
- 7: Send data () `onlyOwner` $\triangleright$ Event triggering (callback function in $S_{contract}$ to notify it about the availability of the requested data).

Phase I: To request data from off-chain to on-chain, pull-outbound oracles are used. To ensure access control, only the contract owner (`msg.sender`) can initiate a pull-outbound oracle request to a deployed $S_{contract}$ once received from Off_{chain} state. This check is essential for verifying the identity of the contract owner and to ensure that the operations of the deployed $S_{contract}$ are restricted to being executed through off-chain states via oracles requests. We use TLSNotary-based authenticity proofs to ensure request integrity and source authenticity. TLSNotary is a cryptographic protocol designed to provide verifiable proof that a specific piece of data was securely obtained from a particular server at a specific time, without exposing or compromising the integrity and confidentiality of the data. Finally, an event register is requested on the blockchain.

Phase II: The system verifies data by validating digital signatures against TLSNotary-based authenticity proofs. If the verification is successful, the data is then formatted according to the required specifications. Following this, a data send event (`Send data ()`) is triggered, which can only be initiated by the contract owner. Once these steps are

⁴ Solidity is a programming language for writing smart contracts for the Ethereum blockchain: <https://soliditylang.org/>.

completed successfully, the $Of f_{chain}$ system receives the data from the On_{chain} source.

4.4. DT services

Algorithm 4: Algorithm 4 outlines the steps involved in utilizing DT services. First, derive a DT instance ($Instance$) from DT_i^{model} . Second, load the data into the derived $Instance$ by calling Phase II: Algorithm 1 (case IoT Hub). Third, specify DT_{PK} which may include χ_{specs} for instance, defining temperate sensor threshold values.

Algorithm 4 DT services: visualization and querying

Input: DT_{MODEL} , $Instance$ $LOAD\_DTDATA(f_{s_i}^{op})$

Output: $Results$ (depending on $Query$)

- 1: Derive $Instance$ from DT_{MODEL} and load to the simulator
 - 2: Call $LOAD\_DTDATA(f_{s_i}^{op})$ from Algo. 1 $\triangleright$ to load data into $Instance$
 - 3: Specify $DT_{PK} \triangleright$ Use supporting functions such as APIs & libraries.
 - 4: Define SQL $Query$ $\triangleright$ Listings 1 and 2
 - 5: Display $Results$ through ADT Explorer
-

To visualize the DT models of robotic arms, use the ADT explorer, which allows users to view, query, and update DT data. The ADT explorer provides structured query language (SQL) support for querying the state of the DT, which reflects the physical asset. For example, we can use a query (see Listing 1) to obtain created DTs, their associated relationships, and their current state based on the defined properties. Moreover, we can perform complex operations to query or make informed decisions. For example, we use a query (see Listing 2) to analyze the robotic arm where the temperature alert is high (Algorithm 4: step 4). This information allows plant operators to cross-check and ensures compliance with the S&S rules.

```
1 SELECT * FROM DIGITALTWINS
```

Listing 1: Excerpt of querying all DT instances.

```
1 SELECT * FROM DIGITALTWINS WHERE IS_OF_MODEL
   (dtmi:robot:digitalroboticarm:
    temperature_sensor;1') AND
   TemperatureAlert = 'High')
```

Listing 2: Excerpt of querying a DT instance with high temperature alert.

We make a general remark that for the sake of our proof of concept, we used the available platform, i.e., ADT; however, any home-grown DT implementation with added functionality can be used with the TRIPLE framework. As there are multiple ways of creating DT models, and there is no singular method to create a DT; therefore, we used ADT to showcase the feasibility of the TRIPLE framework. Furthermore, the proof of concept is limited to a specific use case, i.e., an automotive assembly line. Thus, the results may not be generalized to other CPS systems, and the security measures applied to this use case may not be sufficient for different systems.

5. Formal analysis

Formal analysis is a process of proving the accuracy of a fundamental system by assessing interconnected components and processes, evaluating information flow, and illustrating information processing [56]. Through formal modeling and verification, we demonstrate the correctness of the models based on desirable properties such as conformance to S&S rules and the trustworthiness of data and sources for the TRIPLE framework. We address both security modes of DTs: in replication mode, we verify the data synchronization module, while in simulation mode, we evaluate the impact of sensors, machines, and S&S rules

on the overall model settings. We performed formal verification using satisfiability modulo theories (SMT) and Z3 Solver. The SMT is an area of the automated deduction for checking the satisfiability of formulas over some theories of interest, and has its roots in boolean satisfiability solvers (SAT) [57]. Note that among multiple solvers that support SMT-LIB, we used the Z3 solver- an automated satisfiability checker in our study. Readers are encouraged to refer to [58] for detailed information on the workings and commands of the Z3 solver.

In the following, we formally model (Section 5.1) and verify (Section 5.2) the TRIPLE framework. Table 5 summarizes the notations used in this section.

5.1. Formal modeling of the TRIPLE framework

We utilize high-level petri nets (HLPN) for formal TRIPLE framework modeling [59]. HLPN models consist of *places* (shown as circles and represent data stores), *transitions* (shown as filled elongated rectangles and represent operations on these places), and *flows* (shown as directed arrows and represent the flow of information) as depicted in Fig. 9. Readers are encouraged to read [57,60] for further details on HLPN.

Fig. 9 shows the HLPN model of the TRIPLE framework. Based on the TRIPLE framework (discussed in Section 3), we define formulas (preconditions and postconditions) that map to mathematical transitions. The mapping of places ($\mathbb{P}$) to the types (attributes) is presented in Table 6. This mapping demonstrates how the various components of the model correspond to specific attributes and transitions within the TRIPLE framework. For instance, ' $\Phi(\text{Sensor})$ ' can be interpreted as follows. ' Φ ' is a mapping function for a place 'Sensor' that holds the mappings of sensor ID (S_{ID}), timestamped sensor data (S_t), threshold values defined for sensor (τ_{min}, τ_{max}), current value ($Observed\_Val$), and MAC address of sensor (MAC_addr). $\times$ represents aggregation. Eqs. (1) through (11) list all transitions mapped to these rules. For instance, Eq. (1) illustrates the registration process of a sensor (Reg_S). The flow $x1$ represents the respective attributes originating from the 'sensor', which are necessary to register a sensor or node to the blockchain (see Fig. 9). Note that by following transitions and flows in Fig. 9, other equations can be interpreted in a similar manner.

To formally verify TRIPLE, we consider an assembly line in the automotive industry (refer to case study in Section 4.1: Fig. 5). Before initiating the production process, the participating entities including sensors, e.g., temperature (Sensor5), and machines/devices, e.g., robotic arm are registered (Eqs. (1) and (2)) on the blockchain along with their IDs and other configuration parameters (as discussed in Section 3).

$$\begin{aligned} \mathbf{R}(\text{Reg\_S}) &= \forall X1 \in x1 \wedge \forall X3 \in x3 \mid X3[1] \\ &:= X1[1] \wedge X3[3] \\ &:= X1[3] \wedge X3[4] := X1[4] \wedge X3[5] := X1[6] \\ &\wedge x3' := x3 \cup \{X3[1], X3[3], X3[4], X3[5]\} \end{aligned} \quad (1)$$

$$\begin{aligned} \mathbf{R}(\text{Reg\_M}) &= \forall X2 \in x2 \wedge \forall X4 \in x4 \mid X4[2] := X2[1] \wedge X4[5] \\ &:= X2[4] \wedge x4' := x4 \cup \{X4[2]\} \end{aligned} \quad (2)$$

During the production process, data from registered entities (Reg_S and Reg_M) is input into the data integration and interoperability module (Eqs. (3) and (4)) to fuse sensor-based data and ensure conformance to integrity checks. Examples include monitoring the velocity of the conveyor belt (Sensor1) and controlling the number of chassis loaded

Table 5

Summary of additional notations used in the formal analysis (Section 5). For other notations, refer to Table 2.

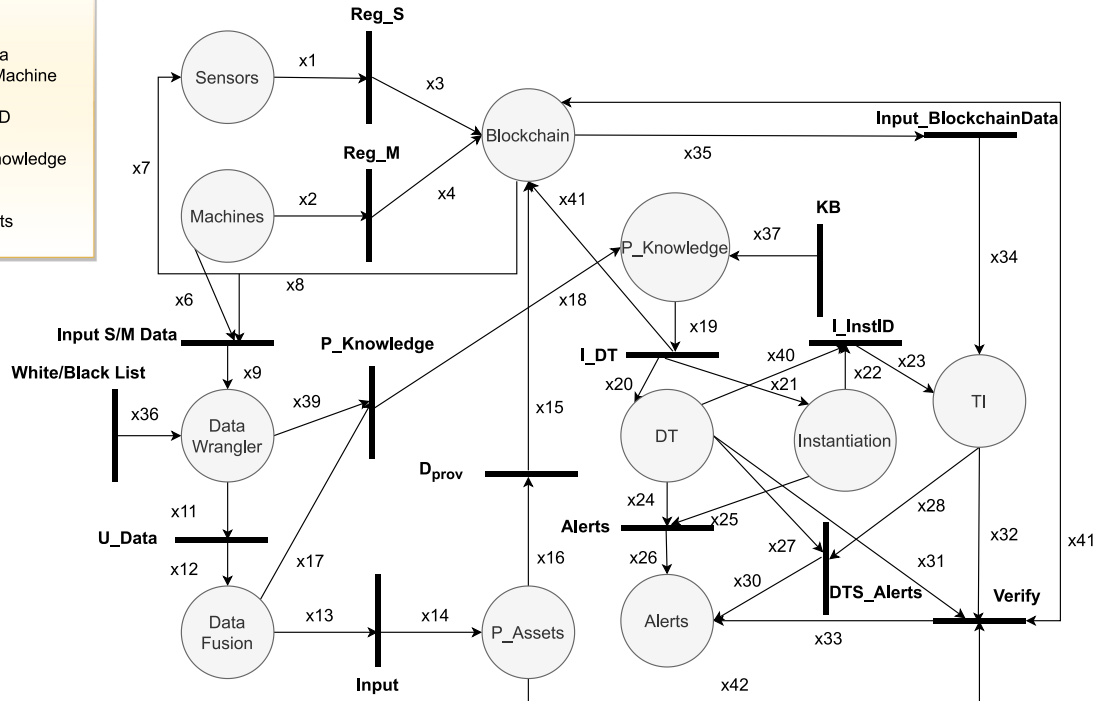
Notation	Explanation	Notation	Explanation
S_i	Timestamped sensor data	W/B_List	White/Black list
M'_{cs}	Timestamped machine current state	M'_s	Timestamped machine state
DT'_{cs}	Timestamped DT current state	DT'_s	Timestamped DT state
S'_d	Timestamped sensor data	Ins/Rep_{ID}	Instance/Replication ID
AT_{type}	Alert type	$ACategory$	Alert category
$Observed\_Val$	Observed value	MAC_addr	Media Access Control (MAC) address
Reg_S	Register sensor	Reg_M	Register machine
I_InstID	Input Instantiation ID	I_DT	Instantiate DT
$P\_Knowledge$	Process knowledge	U_Data	Unified data

Table 6

Mapping of places (including data stores, transitions, and data flows) to the data types (attributes). Refer to Tables 2 and 5 for notations.

Places	Mapping	Description
$\Phi(\text{Sensors})$	$\mathbb{P}(S_{ID} \times S_i \times \tau_{min} \times \tau_{max} \times Observed\_Val \times MAC\_addr)$	Holds sensor-related information, including unique id, timestamped data, predefined threshold values, observed value, and MAC address.
$\Phi(\text{Machines})$	$\mathbb{P}(M_{ID} \times M'_s \times M'_{cs} \times MAC\_addr)$	Holds machine-related information, including unique id, timestamped machine current state, timestamped machine state, and MAC address.
$\Phi(\text{Data Wrangler})$	$\mathbb{P}(S_{ID} \times S_i \times M'_s \times M'_{cs} \times MAC\_addr)$	Holds data from physical assets (sensors and machines).
$\Phi(\text{Data Fusion})$	$\mathbb{P}(S_{ID} \times M_{ID} \times DT_{type} \times D_{\eta})$	Holds fused structured data from the sensor and the machine.
$\Phi(\text{Blockchain})$	$\mathbb{P}(S_{ID} \times M_{ID} \times \tau_{min} \times \tau_{max} \times MAC\_addr \times W\_List \times B\_List \times D_{prov} \times SS)$	Holds critical data and provenance data.
$\Phi(\text{P\_Asset})$	$\mathbb{P}(S_{ID} \times M_{ID} \times DT'_{cs} \times MAC\_addr \times W\_List \times B\_List)$	Holds asset-related information.
$\Phi(\text{P\_Knowledge})$	$\mathbb{P}(S_{ID} \times M_{ID} \times \tau_{min} \times \tau_{max} \times MAC\_addr \times W\_List \times B\_List \times KB \times ML)$	Represents information contained in the knowledge base.
$\Phi(\text{DT})$	$\mathbb{P}(Rep_{ID} \times M_{ID} \times DT'_s \times S_{ID} \times S'_d)$	Represents DT-related information.
$\Phi(\text{Instantiation})$	$\mathbb{P}(S_{ID} \times M_{ID} \times Ins_{ID} \times Rep_{ID})$	Holds information about all DT instances.
$\Phi(\text{Alerts})$	$\mathbb{P}(Rep_{ID} \times AT_{type} \times ACategory)$	Provides information about the types of alerts generated in the system.
$\Phi(\text{TI})$	$\mathbb{P}(S_{ID} \times M_{ID} \times Ins_{ID} \times Rep_{ID} \times SS)$	Represents the threat intelligence achieved through the safety and security rules.

S/M: Sensor / Machine
U_Data: Unified Data
Prov_Data: Provenance Data
Reg_S/M: Register Sensor/Machine
I_DT: Instantiate DT
I_InstID: Input Instantiation ID
KB: Knowledge Base
P_Knowledge: Process Knowledge
TI: Threat Intelligence
DT: Digital Twin
DTS_Alerts: DT Sensor Alerts
P_Assets: Physical Assets

**Fig. 9.** High-level Petri Net (HLPN) model of the TRIPLE framework. Circles represent places, elongated rectangles represent transitions, and directed arrows represent flows. $x(\text{number})$ represents a transition number across places and transitions.

onto the belt (Sensor2) to prevent collisions between adjacent chassis on the assembly line.

$$\begin{aligned}
 \mathbf{R}(\text{Input S/M Data}) &= \forall X6 \in x6 \wedge \forall X7 \in x7 \wedge \forall X8 \in x8 \wedge \forall X9 \\
 &\in x9 \mid X9[1] := X7[1] \wedge X9[2] \\
 &:= X6[1] \wedge X9[3] := X7[2] \wedge X9[4] \\
 &:= X7[3] \wedge X9[5] := X7[4] \wedge X9[6] \\
 &:= (X7[6] \vee X6[4]) \wedge X9[9] \\
 &:= X7[2] \wedge X9[10] := X6[2] \wedge x9' := x9 \cup \{X9[1], X9[2], X9[3], X9[4], \\
 &\quad X9[5], X9[6], X9[9], X9[10]\}
 \end{aligned} \tag{3}$$

$$\begin{aligned}
 \mathbf{R}(\text{U\_Data}) &= \forall X11 \in x11 \wedge \forall X12 \in x12 \mid X12[1] \\
 &:= X11[1] \wedge X12[2] := X11[2] \wedge X12[4] \\
 &:= \text{Unify}(X9[9], X9[10])x12' := x12 \cup \{X12[1], X12[2], X12[4]\}
 \end{aligned} \tag{4}$$

For tracking data and data generating sources, the hash of provenance data (D_{prov}) is stored in blockchain (Eq. (5)).

$$\begin{aligned}
 \mathbf{R}(\text{D}_{prov}) &= \forall X16 \in x16 \wedge \forall X15 \in x15 \mid X15[8] = \\
 \text{Provenance}(X16[3])_{\forall X16[1]=X15[1] \wedge \forall X16[2]=X15[2]} x15' &:= x15 \cup \{X15[8]\}
 \end{aligned} \tag{5}$$

To model DTs, unified data (U_Data) from the data fusion method is inputted to process knowledge ($P\_Knowledge$) as defined in Eq. (6). Based on $P\_Knowledge$, the replication and simulation modes of DTs are modeled (Eq. (7)).

$$\begin{aligned}
 \mathbf{R}(\text{P\_Knowledge}) &= \forall X17 \in x17 \wedge \forall X18 \in x18 \wedge \forall X39 \in x39 \mid X18[1] \\
 &:= X17[1] \wedge X18[2] := X17[2] \wedge \\
 &\quad X18[3] := X39[4] \wedge X18[4] := X39[5] \wedge X18[5] \\
 &:= X39[6] \wedge X18[6] := X39[7] \wedge X18[7] := X39[8] \wedge \\
 &\quad x18' := x18 \cup \{X18[1], X18[2], X18[3], X18[4], X18[5], X18[6], X18[7]\}
 \end{aligned} \tag{6}$$

$$\begin{aligned}
 \mathbf{R}(\text{I\_DT}) &= \forall X19 \in x19 \wedge \forall X20 \in x20 \wedge \forall X21 \in x21 \wedge \\
 &\quad \forall X41 \in x41 \mid \text{Get\_Config}(X19[1] \\
 &\quad \wedge X19[2] \wedge X19[3] \wedge X19[4] \wedge X19[5] \wedge X19[6] \wedge X19[7] \\
 &\quad \wedge X19[8] \wedge X19[9] = \text{True}) \\
 &\rightarrow (X20[1] := \text{Create/Update\_DT}(\text{Get\_Config}) \wedge X20[2] \\
 &\quad := X19[2] \wedge X20[5] := S_{D_r}(X19[1]) \\
 &\quad \wedge X20' := x20 \cup \{X20[1], X20[2], X20[3], X20[4], X120[5]\}) \\
 &\quad \times (\text{Create\_Insta} = \text{True} \rightarrow X21[1] := X19[1] \\
 &\quad \wedge X21[2] := X19[2] \wedge X21[3] := (\text{Create\_IID} \wedge X21[4] := X20[1] \wedge x21' \\
 &\quad := x21 \cup \{X21[1], X21[2], X21[4]\}) \\
 &\quad X41[9] := \text{Gen\_S\&S}(X20[1])_{(\forall X20[1]=X41[1] \vee \forall X20[2]=X41[2]) \wedge X21[3] \neq X41[1]} \\
 &\quad \wedge x41' := x41 \cup \{X41[9]\})
 \end{aligned} \tag{7}$$

TI module takes the data from DT (I_DT) and Instantiation (I_InstD) along with the processed knowledge ($P\_Knowledge$) and auxiliary information from the blockchain ($Input\_BlockchainData$) to detect failures and call the corresponding scheduling services or model calibration services in the virtual space to carry out the necessary measures such as configuring the machine settings (as Eqs. (8)–(11) shows). For example, in ICS, if the temperature of the welding arm crosses the threshold, then the TI module will generate an alert and call the corresponding service module. Similarly, if any failures or mismatched configurations are detected by the TI module (see $\mathbf{R}(\text{Verify})$:

Eq. (11)), then it also generates an alert to take necessary actions. The alerts ($AType$) could be of different types ($ACategory$), i.e., critical alerts (such as low, medium, and high priority), and non-critical alerts (such as warning, caution, and notification). For instance, in the case of ICS (see Fig. 5), if the welding arm's temperature is a few degrees less than the maximum threshold ($Expected\_Thrs$), then a warning alert can be generated. Similarly, if the temperature crosses the maximum threshold, then a high-priority alert can be generated to inform the operator. To ensure data integrity and availability, essential data, e.g., models and S&S rules are stored and retrieved from the blockchain ($Input\_BlockchainData$) (Eq. (9)).

$$\begin{aligned}
 \mathbf{R}(\text{I\_InstD}) &= \forall X22 \in x22 \wedge \forall X23 \in x23 \wedge \forall X40 \in x40 \mid X23[1] \\
 &:= X22[1] \wedge X23[2] := \\
 &\quad X22[2] \wedge X23[3] := X22[3] \wedge X23[4] := X22[4] \wedge x23' \\
 &:= x23 \cup \{X23[1], X23[2], X23[3], X23[4]\}
 \end{aligned} \tag{8}$$

$$\begin{aligned}
 \mathbf{R}(\text{Input\_BlockchainData}) &= \forall X35 \in x35 \wedge \forall X34 \in x34 \mid \\
 &\quad X34[5] := X22[9]_{\forall X34[1]=X35[1] \vee \forall X34[2]=X35[2]} \wedge := x34' \\
 &:= x34 \cup \{X34[5]\}
 \end{aligned} \tag{9}$$

$$\begin{aligned}
 \mathbf{R}(\text{Input}) &= \forall X13 \in x13 \wedge \forall X14 \in x14 \mid X14[1] := X13[1] \wedge X14[2] \\
 &:= X13[2] \wedge X14[3] \\
 &:= C\_State(X13[1], X13[2]) \wedge X14[4] := X13[6] \wedge X14[5] \\
 &:= X13[7] \wedge X14[6] := X13[8] \wedge x14' := x14 \cup \{X14[1], X14[2], X14[3], \\
 &\quad X14[4], X14[5], X14[6]\}
 \end{aligned} \tag{10}$$

$$\begin{aligned}
 \mathbf{R}(\text{Verify}) &= \forall X31 \in x31 \wedge \forall X32 \in x32 \wedge \forall X41 \in x41 \wedge \forall X42 \in x42 \\
 &\quad \wedge \forall X33 \in x33 \mid \\
 &\quad X31[3] \neq X42[3] \vee (X31[2] \vee X31[4]) \in X41[6] \vee (X41[3] \vee X41[4]) \\
 &\quad \neq Expected\_Thrs_{\forall X31[1] \neq (X42[1] \vee X41[2])} \rightarrow X33[1] := X31[1] \wedge \\
 &\quad X33[2] := AType \wedge X33[3] := ACategory(X33[2]) \wedge x33' := x33 \\
 &\quad \cup \{X33[1], X33[2], X33[3]\}
 \end{aligned} \tag{11}$$

5.2. Formal verification of the TRIPLE framework

To formally verify the TRIPLE framework, the HLPN model is first translated into the satisfiability modulo theories library (SMT-Lib). Then, the models, along with the properties, are provided to the Z3 solver, which checks if the properties are satisfied by the models or not. Our model adopts two security evaluation processes, (a) replication and (b) simulation. For replication mode, we choose to verify the data sync module of the TRIPLE framework. The property of data consistency holds if the model can detect inconsistencies based on the mapping of physical and twin states. For simulation mode, we seek to simulate the effect of a particular object (sensor, machine/device, S&S rule, or any other attribute) on the overall settings of the model. Therefore, we chose different sensors and machines, included them in an existing model, and then verified the stated property. We deliberately introduced some inconsistent values to evaluate the correctness of our model.

Table 7 presents the verification results and the time taken by the Z3 solver for varying numbers of DT instances in both replication and simulation modes. To evaluate the model's performance, we introduce inconsistencies and/or misconfigurations into the DT instances during each iteration. This approach helps validate whether the model can detect these inconsistencies or misconfigurations. The execution time for property verification in simulation mode is notably high due to the number of inconsistencies introduced. Once the solver identifies an inconsistency and confirms that the property is satisfied, it ceases further

Table 7
Verification time to identify inconsistencies in DT instance (s).

DT instance(s)/#of inconsistencies (Replication, Simulation)	Inconsistency	Execution time (msec)	
		Replication	Simulation
10/(2,3)	Sat	5	3.93
20/(7,5)	Sat	13.58	14.13
30/(11,9)	Sat	19.43	20.42
40/(15,13)	Sat	23.05	25.97
50/(21,19)	Sat	27.85	29.06

execution of that instance and proceeds to the next one. Therefore, the larger the number of inconsistencies, the lower will be the execution time. The verification result ‘Sat’ represents that the solver identified the combination for the values of variables that satisfy the stated property. The execution time appears to scale linearly with the number of DT instances, which confirms TRIPLE’s scalability. The proposed model successfully captures the inconsistencies and/or misconfigurations in the DT instance(s) at every iteration of the verification process, which proves the reliable working of the model.

6. Discussion

Leveraging DTs for CPS security is ambitious yet complex. In this paper, we envision a framework that captures the security requirements of the CPS and try to provide an answer to the following question: *how can DTs help in addressing the security challenges of not only the CPS but CPS-DT lifecycle?*

The confluence of blockchain with DTs for CPS security is fueled by the need for integrity, immutability, and transparency of the data injected into DTs for intelligent and informed decisions. Traditional storage solutions such as databases alone do not guarantee these aspects [61]. Therefore, databases could be used in harmony with the blockchain. It is also worth mentioning that DT alone cannot address CPS security requirements and challenges. Therefore, our framework is based on the harmony among different modules such as TI and blockchain working together with DTs. Additionally, it is also important to note that if DTs are leveraged as security-enhancing capabilities for CPS, then the security of DTs is also of paramount importance as also discussed in [4,62]. Therefore, the TRIPLE framework focuses on the security of the CPS-DT lifecycle.

In the following, we conclude the TRIPLE framework by discussing its impact (Section 6.1), its role as a foundational framework (Section 6.2), lesson learned (Section 6.3), and its limitations (Section 6.4).

6.1. Practical impact of TRIPLE framework

In the wake of cyberattacks targeting manufactured products, manufacturing processes, or entire manufacturing ecosystems [1], there is a critical need to implement proactive security mechanisms to safeguard CPS infrastructure. Therefore, reactive security solutions might not suffice for CPS security. TRIPLE proposed incorporating blockchain, DT, and TI, where blockchain acts as a trustworthy data acquisition and dissemination platform to the CPS-DT ecosystem, DT provides a virtual environment to analyze and assess cascading failures or inconsistencies, and TI proactively supports prediction capabilities in the DTs and eventually to the physical system.

6.2. TRIPLE as a modular framework

One of the other important points to note is the realization of the DTs. To date, there is still a lack of standards for DTs, and thus the implementation and realization of DTs are subjective and depend on the available resources. However, there is a consensus on the baseline required components. For instance, DTs will need considerable

computing, storage, and communication capabilities. The boundary between physical assets and DTs is also important to consider for the DTs realization. To be more precise, the creation, deployment, and operation of DTs can be realized through various technologies and infrastructures, such as the Internet of Things (IoT), AI/ML, cloud-fog-edge computing, virtualization, communication networks (5G and Beyond), storage, software-defined networks, data analytics, and visualization [4,63]. The administrative domain boundaries and physical boundaries between CPS and DTs, and the availability of the aforementioned resources are important factors to consider for the realization of DTs. Therefore, we kept the TRIPLE framework as abstract as possible to incorporate most of the above requirements and it can be adopted in a specific environment. Furthermore, in this paper, we focused on the baseline validity of the TRIPLE. Although TRIPLE is a modular framework; however, for the sake of PoC, we have used specific parameters to showcase the feasibility of the TRIPLE framework and these parameters may produce different results with different settings such as blockchain platform, consensus algorithm, and network size. The reason is to put forth a modular framework that could be further investigated for specific use cases as well as available solutions.

Fig. 10 illustrates the realization of the TRIPLE framework as a modular framework. Depending on the underlying use case, security objectives, and available resources (both capital and operational), industries can adapt the TRIPLE framework by selecting a blockchain platform that meets their privacy, confidentiality, and access control requirements. Fig. 10 also highlights that DT modeling parameters, such as virtual representation’s fidelity, need to be considered; higher fidelity incurs greater costs but is more suitable for replication operation modes [64]. Additionally, the choice to incorporate TI into the framework is influenced by the organization’s ability to allocate resources for executing proactive or reactive threat hunting, which in turn depends on the specific use case and security objectives.

6.3. Lesson learned

In the following, we discuss the key takeaways, for instance, security challenges in DTs (Section 6.3.1) and the relevancy of the blockchain platform (Section 6.3.2).

6.3.1. Security challenges in DTs

The confluence of technologies and interaction between physical, cyber, and cyber-physical domains opens a variety of security threats [4]. It raises questions about the security-enhancing (c)ability of DTs. For example, the issue of fidelity presents a challenge. While high-fidelity DTs are valuable for effectively testing the security of CPSs in a virtual environment, they may also expose critical infrastructure details to sophisticated attackers targeting the DTs [2,4]. Consequently, it is essential to find an optimal balance between sufficient DT fidelity, maximized utility, and the security of the CPS [64]. It is worth noting that fidelity in DTs is subjective, and the required level of fidelity cannot be generalized when designing DTs [20,42].

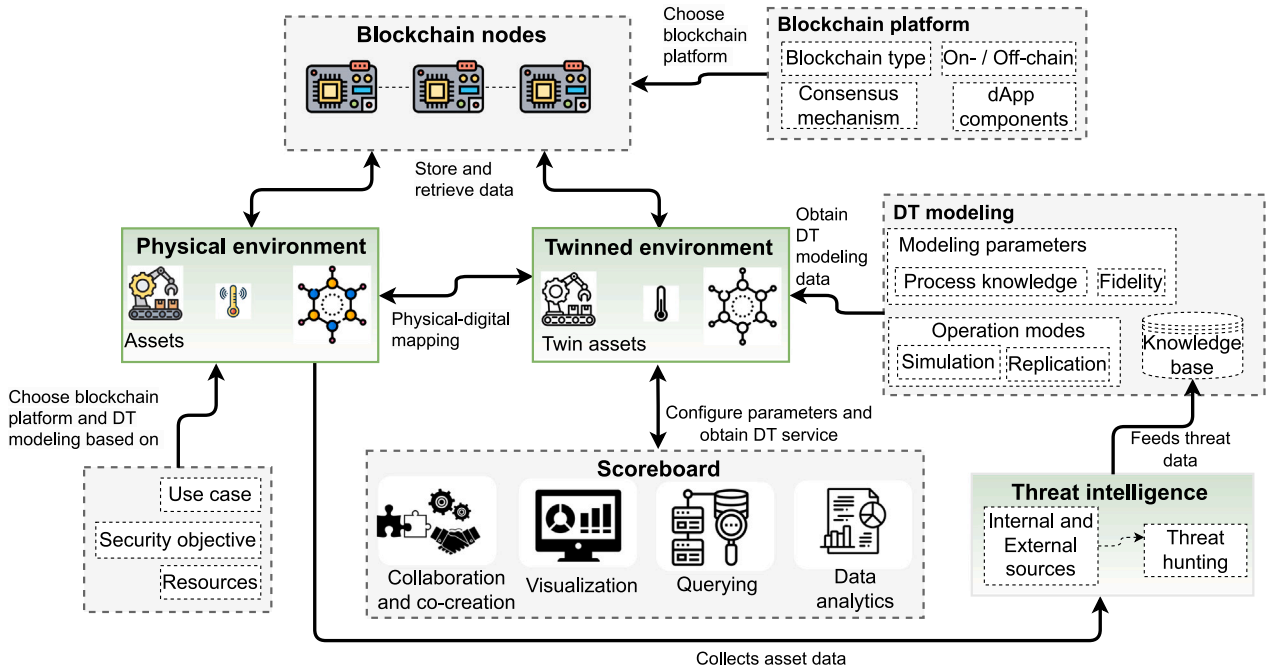


Fig. 10. Realization of the TRIPLE as a modular framework.

6.3.2. Relevancy of blockchain platform

According to the model of Wust et al. [55], for a given use case, the type of blockchain (permissionless, public, and private permissioned) to be used, is based on factors including, but not limited to, the number of readers/writers of the data, presence of a trusted third party, identification of readers/writers, and the nature of verifiability of the exchanged data [55] or more specifically based on the application-specific requirements. Therefore, we can conclude that addressing privacy leakage is subject to the use case and is not necessarily mandatory for all applications. For instance, blockchain-based DTs in healthcare are more sensitive to privacy concerns [65]. In the TRIPLE framework, privacy, if required, can be enforced based on the chosen blockchain platforms, which allow for tailored access controls and permissions. Additionally, privacy can be controlled by leveraging smart contract-based decentralized role-based access controls for automating and enforcing privacy policies, and by using zero-knowledge proofs to verify data without disclosing the data itself [66]. Similarly, confidentiality in the TRIPLE framework depends on the specific use case. To enforce confidentiality, sensitive data can be encrypted and stored off-chain, with only the hash of the encrypted data recorded on-chain.

6.4. Limitations

We identify the following limitations of the TRIPLE framework. *First*, we cannot derive definitive conclusions without conducting experiments on a real automotive assembly line. Therefore, further investigation is necessary to assess the practicality of the concept in real-world settings. *Second*, due to the absence of self-configuring cognitive DTs that support dynamic settings [67], the TRIPLE framework is more suitable for physical environments where production configurations remain static, such as traditional manufacturing lines, which may limit its applicability in more dynamic and rapidly changing settings such as autonomous vehicle networks.

7. Conclusion and future work

In this work, we present **TR**usted and **I**ntelligent cyber-**P**hysical **s**ystEm (TRIPLE) framework (see Fig. 2) that leverages DTs, blockchain, and TI for securing CPSs. We anticipate that the TRIPLE framework can serve as a modular framework for CPS security, particularly in industrial manufacturing scenarios with static production configurations, where equipment, processes, and workflows are stable and infrequently altered. Moreover, within ICS use cases, it allows for flexible implementation based on (a) specific use cases, (b) objectives, and (c) available resources (see Fig. 10), thereby allowing the manufacturing plant operators to choose DT characteristics (such as medium to high fidelity), blockchain platforms (such as permissionless, public, and private permissioned) and TI (proactive or reactive) solutions. We demonstrate the application of the TRIPLE framework in the assembly line of the automotive industry (Section 4.1) through a preliminary proof of concept (Section 4). We provide the implementation details (refer to Appendix B) for designing and deploying DT (Section 4.2) in Microsoft ADT, and integration of the Ethereum blockchain (Section 4.3). The former ensures *security through DT* whereas the latter ensures the *security of CPS and DTs* (Section 4.3). Additionally, we illustrate how to view and query DT instances through ADT Explorer (Section 4.4). Furthermore, we perform formal analysis (Section 5) of the TRIPLE framework to demonstrate the correctness of the models based on desirable properties such as conformance to rules and the trustworthiness of data and data-generating sources.

A promising direction for future work is incorporating industry solutions to ensure the TRIPLE framework's applicability in practical industry settings. This involves considering guidelines such as those outlined in [68], which address various facets including design, development, deployment, and operation of DTs. Furthermore, a comprehensive security risk assessment of the TRIPLE framework within the context of the use case should be conducted to identify potential

vulnerabilities, assess the impact of identified threats, and propose mitigation strategies to enhance the overall resilience and security of the framework.

CRedit authorship contribution statement

Sabah Suhail: Conceptualization, Data curation, Investigation, Methodology, Validation, Visualization, Writing – original draft, Writing – review & editing. **Mubashar Iqbal:** Conceptualization, Data curation, Investigation, Methodology, Validation, Visualization, Writing – original draft, Writing – review & editing. **Rasheed Hussain:** Conceptualization, Data curation, Investigation, Methodology, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Saif Ur Rehman Malik:** Conceptualization, Data curation, Formal analysis, Validation, Visualization, Writing – original draft. **Raja Jurdak:** Conceptualization, Data curation, Investigation, Methodology, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partly supported by the Centre for Secure Information Technologies, EPSRC ref. EP/X022323/1.

Appendix A. Listings

```
1 HostName={IoTHubName}.azure-devices.net;
2   SharedAccessKeyName=iothubowner;
3   SharedAccessKey={hubSharedAccessKey}
4   //Connecting with Azure IoT Hub through
   $iothubowner$ access rights
```

Listing 3: Excerpt of Azure IoT Hub connection string.

```
1 HostName={IoTHubName}.azure-devices.net;
2   DeviceId={deviceName};
3   SharedAccessKey={deviceSharedAccessKey}
4   //Connection with an IoT device to gain
   access to read and write data from the
   IoT device
```

Listing 4: Excerpt of registered IoT device connection string.

Appendix B. GitHub repository

The source code can be found at <https://github.com/mubashar-iqbal/triple-digital-twin>. The source code includes two models (RobotcArmInterface.json, TemperatureSensor.json) that have properties (e.g., Id, RoboticArmLocation, Temperature, TemperatureAlert, DeviceState).

Appendix C. Glossary of terms

- *Process knowledge* is the set of rules, connections, flows, states, and data involved in the system for which the DT is modeled, and it collectively contributes to the fidelity of the DT [19].
- *Knowledge base (KB)* comprises lifecycle data, domain data, and contextual knowledge necessary for DT modeling.
- *DT instance* is an instantiation of a DT (either in full or part).
- *DT simulation* operates in an isolated virtual environment without having a direct connection to the live systems [8].
- *DT replication* operates with having a continuously connected and synchronized connection to the live systems [21].
- *Fidelity* in DT is the degree of resemblance between a physical asset and its twin counterpart in terms of asset data, state, behavior, and the frequency of updates as well as the level of abstraction [20].

Data availability

No data was used for the research described in the article.

References

- [1] M.H. Rahman, T. Wuest, M. Shafae, Manufacturing cybersecurity threat attributes and countermeasures: Review, meta-taxonomy, and use cases of cyberattack taxonomies, *J. Manuf. Syst.* 68 (2023) 196–208, <http://dx.doi.org/10.1016/j.jmsy.2023.03.009>.
- [2] M. Eckhart, A. Ekelhart, Digital Twins for Cyber-Physical Systems Security: State of the Art and Outlook, Springer International Publishing, 2019, pp. 383–412, http://dx.doi.org/10.1007/978-3-030-25312-7_14.
- [3] S. Suhail, R. Hussain, R. Jurdak, A. Oracevic, K. Salah, C.S. Hong, R. Matulevicius, Blockchain-based digital twins: Research trends, issues, and future challenges, *ACM Comput. Surv.* 54 (11s) <http://dx.doi.org/10.1145/3517189>.
- [4] C. Alcaraz, J. Lopez, Digital twin: A comprehensive survey of security threats, *IEEE Commun. Surv. Tutor.* 24 (3) (2022) 1475–1503, <http://dx.doi.org/10.1109/COMST.2022.3171465>.
- [5] M. Dietz, G. Pernul, Unleashing the digital twin's potential for ics security, *IEEE Secur. Priv.* 18 (4) (2020) 20–27, <http://dx.doi.org/10.1109/MSEC.2019.2961650>.
- [6] Industroyer2: Industroyer reloaded, ESET research, 2022, Available at: <https://www.welivesecurity.com/2022/04/12/industroyer2-industroyer-reloaded/>, (Accessed 15 May 2022).
- [7] A. Cherepanov, Win32/industroyer: A new threat for industrial control systems, 2017, Available at: https://web-assets.esetstatic.com/wls/2017/06/Win32_Industroyer.pdf, (Accessed 14 April 2022).
- [8] M. Eckhart, A. Ekelhart, Towards security-aware virtual environments for digital twins, in: Proceedings of the 4th ACM Workshop on Cyber-Physical System Security, CPSS '18, Association for Computing Machinery, 2018, pp. 61–72, <http://dx.doi.org/10.1145/3198458.3198464>.
- [9] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, K.-Y. Tung, Review: Intrusion detection system: A comprehensive review, *J. Netw. Comput. Appl.* 36 (1) (2013) 16–24, <http://dx.doi.org/10.1016/j.jnca.2012.09.004>.
- [10] G. Kavallieratos, S.K. Katsikas, V. Gkioulos, Towards a cyber-physical range, CPSS '19, Association for Computing Machinery, 2019, pp. 25–34, <http://dx.doi.org/10.1145/3327961.3329532>.
- [11] R. Bitton, et al., Deriving a cost-effective digital twin of an ICS to facilitate security evaluation, in: European Symposium on Research in Computer Security, Springer, 2018, pp. 533–554, http://dx.doi.org/10.1007/978-3-319-99073-6_26.
- [12] R. Minerva, N. Crespi, Digital twins: Properties, software frameworks, and application scenarios, *IT Prof.* 23 (1) (2021) 51–55, <http://dx.doi.org/10.1109/MITP.2020.2982896>.

- [13] M. Eckhart, et al., Security-enhancing digital twins: Characteristics, indicators, and future perspectives, *IEEE Secur. Priv.* 21 (6) (2023) 64–75, <http://dx.doi.org/10.1109/MSEC.2023.3271225>.
- [14] F. Tao, M. Zhang, Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing, *IEEE Access* 5 (2017) 20418–20427, <http://dx.doi.org/10.1109/ACCESS.2017.2756069>.
- [15] D. Hadžiosmanović, R. Sommer, E. Zambon, P.H. Hartel, Through the eye of the plc: Semantic security monitoring for industrial processes, in: *Proceedings of the 30th Annual Computer Security Applications Conference, ACSAC '14*, Association for Computing Machinery, 2014, pp. 126–135, <http://dx.doi.org/10.1145/2664243.2664277>.
- [16] A. Dorri, S. Kanhere, R. Jurdak, *Blockchain for Cyberphysical Systems*, Artech House, 2020.
- [17] R. McMillan, Definition: Threat intelligence, 2021, Available at: <https://www.gartner.com/en/documents/2487216>, (Accessed 20 May 2013).
- [18] N. Sun, M. Ding, J. Jiang, W. Xu, X. Mo, Y. Tai, J. Zhang, Cyber threat intelligence mining for proactive cybersecurity defense: A survey and new perspectives, *IEEE Commun. Surv. Tutor.* 25 (3) (2023) 1748–1774, <http://dx.doi.org/10.1109/COMST.2023.3273282>.
- [19] S. Suhail, M. Iqbal, R. Hussain, R. Jurdak, ENIGMA: An explainable digital twin security solution for cyber-physical systems, *Comput. Ind.* 151 (2023) 103961, <http://dx.doi.org/10.1016/j.compind.2023.103961>.
- [20] D. Jones, C. Snider, A. Nassehi, J. Yon, B. Hicks, Characterising the digital twin: A systematic literature review, *CIRP J. Manuf. Sci. Technol.* 29 (2020) 36–52, <http://dx.doi.org/10.1016/j.cirpj.2020.02.002>.
- [21] M. Eckhart, A. Ekelhart, A. specification-based state replication approach for digital twins, in: *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and Privacy, CPS-SPC '18*, Association for Computing Machinery, 2018, pp. 36–47, <http://dx.doi.org/10.1145/3264888.3264892>.
- [22] Sabah Suhail, Saif Ur Rehman Malik, Raja Jurdak, Rasheed Hussain, Raimundas Matulevičius, Davor Svetinovic, Towards situational aware cyber-physical systems: A security-enhancing use case of blockchain-based digital twins, *Comput. Ind.* 141 (2022) 103699, <http://dx.doi.org/10.1016/j.compind.2022.103699>.
- [23] M. Dietz, M. Vielberth, G. Pernul, Integrating digital twin security simulations in the security operations center, in: *Proceedings of the 15th International Conference on Availability, Reliability and Security, ARES '20*, Association for Computing Machinery, 2020, <http://dx.doi.org/10.1145/3407023.3407039>.
- [24] M. Groshev, C. Guimarães, J. Martín-Pérez, A. de la Oliva, Toward intelligent cyber-physical systems: Digital twin meets artificial intelligence, *IEEE Commun. Mag.* 59 (8) (2021) 14–20, <http://dx.doi.org/10.1109/MCOM.001.2001237>.
- [25] Q. Xu, S. Ali, T. Yue, Digital twin-based anomaly detection in cyber-physical systems, in: *2021 14th IEEE Conference on Software Testing, Verification and Validation*, 2021, pp. 205–216, <http://dx.doi.org/10.1109/ICST49551.2021.00031>.
- [26] J. Humble, D. Farley, *Continuous delivery: Reliable software releases through build, in: Test, and Deployment Automation*, 1st ed., Addison-Wesley Professional, 2010.
- [27] I. Kumara, M. Garriga, A.U. Romeu, D. Di Nucci, F. Palomba, D.A. Tamburri, W.-J. van den Heuvel, The do's and don'ts of infrastructure code: A systematic gray literature review, *Inf. Softw. Technol.* 137 (2021) 106593, <http://dx.doi.org/10.1016/j.infsof.2021.106593>.
- [28] A. Patel, T. Schenk, S. Knorn, H. Patzlaff, D. Obradovic, A.B. Halblaub, Real-time, simulation-based identification of cyber-security attacks of industrial plants, in: *2021 IEEE International Conference on Cyber Security and Resilience, CSR*, 2021, pp. 267–272, <http://dx.doi.org/10.1109/CSR51186.2021.9527938>.
- [29] I. Yaqoob, K. Salah, M. Uddin, R. Jayaraman, M. Omar, M. Imran, Blockchain for digital twins: Recent advances and future research challenges, *IEEE Netw.* 34 (5) (2020) 290–298, <http://dx.doi.org/10.1109/MNET.001.1900661>.
- [30] A. Somma, A. De Benedictis, C. Esposito, N. Mazzocca, The convergence of digital twins and distributed ledger technologies: A systematic literature review and an architectural proposal, *J. Netw. Comput. Appl.* 225 (2024) 103857, <http://dx.doi.org/10.1016/j.jnca.2024.103857>.
- [31] M. Dietz, B. Putz, G. Pernul, A distributed ledger approach to digital twin secure data sharing, in: *IFIP Annual Conference on Data and Applications Security and Privacy*, Vol. 11559, Springer, 2019, pp. 281–300, http://dx.doi.org/10.1007/978-3-030-22479-0_15.
- [32] F. Tao, H. Zhang, C. Zhang, Advancements and challenges of digital twins in industry, *Nat. Comput. Sci.* 4 (3) (2024) 169–177, <http://dx.doi.org/10.1038/s43588-024-00603-w>.
- [33] K. Willcox, B. Segundo, The role of computational science in digital twins, *Nat. Comput. Sci.* 4 (3) (2024) 147–149, <http://dx.doi.org/10.1038/s43588-024-00609-4>.
- [34] R.M. Lee, R.T. Lee, *Sans 2018 Threat Hunting Survey Results*, SANS Institute Information Reading Room.
- [35] O. Novo, Blockchain meets IoT: An architecture for scalable access management in IoT, *IEEE Internet Things J.* 5 (2) (2018) 1184–1195, <http://dx.doi.org/10.1109/JIOT.2018.2812239>.
- [36] F. Zafar, et al., Trustworthy data: A survey, taxonomy and future trends of secure provenance schemes, *J. Netw. Comput. Appl.* 94 (2017) 50–68, <http://dx.doi.org/10.1016/j.jnca.2017.06.003>.
- [37] S. Suhail, R. Hussain, R. Jurdak, C.S. Hong, Trustworthy digital twins in the industrial internet of things with blockchain, *IEEE Internet Comput.* 26 (3) (2022) 58–67, <http://dx.doi.org/10.1109/MIC.2021.3059320>.
- [38] L. Durão, S. Haag, R. Anderl, K. Schützer, E. Zancul, Digital twin requirements in the context of industry 4.0, in: *15th IFIP International Conference on Product Lifecycle Management, AICT-540 of Product Lifecycle Management To Support Industry 4.0*, Springer International Publishing, 2018, pp. 204–214, http://dx.doi.org/10.1007/978-3-030-01614-2_19.
- [39] L. Liu, X. Song, C. Zhang, D. Tao, Gan-mdf: An enabling method for multi-fidelity data fusion, *IEEE Internet Things J.* (2022) 1, <http://dx.doi.org/10.1109/JIOT.2022.3142242>.
- [40] A. Rasheed, O. San, T. Kvamsdal, Digital twin: Values, challenges and enablers from a modeling perspective, *IEEE Access* 8 (2020) 21980–22012, <http://dx.doi.org/10.1109/ACCESS.2020.2970143>.
- [41] Y. Wang, Z. Su, S. Guo, M. Dai, T.H. Luan, Y. Liu, A survey on digital twins: Architecture, enabling technologies, security and privacy, and future prospects, *IEEE Internet Things J.* (2023) 1, <http://dx.doi.org/10.1109/JIOT.2023.3263909>.
- [42] A. Arrieta, Multi-fidelity digital twins: a means for better cyber-physical systems testing?, 2021, Available at: <https://arxiv.org/abs/2101.05697>.
- [43] O. Bouachir, M. Aloqaily, L. Tseng, A. Boukerche, Blockchain and fog computing for cyberphysical systems: The case of smart industry, *Computer* 53 (9) (2020) 36–45, <http://dx.doi.org/10.1109/MC.2020.2996212>.
- [44] Y.P. Tsang, C.K.M. Lee, K. Zhang, C.H. Wu, W.H. Ip, On-chain and off-chain data management for blockchain-internet of things: A multi-agent deep reinforcement learning approach, *J. Grid Comput.* 22 (1) (2024) 16, <http://dx.doi.org/10.1007/s10723-023-09739-x>.
- [45] M. Iqbal, R. Matulevičius, Exploring sybil and double-spending risks in blockchain systems, *IEEE Access* 9 (2021) 76153–76177, <http://dx.doi.org/10.1109/ACCESS.2021.3081998>.
- [46] K. Mammadzada, et al., Blockchain oracles: A framework for blockchain-based applications, in: *Business Process Management: Blockchain and Robotic Process Automation Forum*, Springer International Publishing, 2020, pp. 19–34.
- [47] H.Y. Wu, X. Yang, C. Yue, H.-Y. Paik, S.S. Kanhere, Chain or DAG? underlying data structures, architectures, topologies and consensus in distributed ledger technology: A review, taxonomy and research issues, *J. Syst. Archit.* 131 (2022) 102720, <http://dx.doi.org/10.1016/j.sysarc.2022.102720>.
- [48] G. Urdaneta, G. Pierre, M.V. Steen, A survey of DHT security techniques, *ACM Comput. Surv.* 43 (2) <http://dx.doi.org/10.1145/1883612.1883615>.
- [49] E. Sit, R. Morris, Security considerations for peer-to-peer distributed hash tables, in: *Peer-To-Peer Systems*, Springer Berlin Heidelberg, 2002, pp. 261–269.
- [50] A.K. Fedorov, E.O. Kiktenko, A.I. Lvovsky, Quantum computers put blockchain security at risk, *Nature* 563 (2018) 465–467, <http://dx.doi.org/10.1038/d41586-018-07449-z>.
- [51] S. Samtani, R. Chinn, H. Chen, J.F. Nunamaker Jr., Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence, *J. Manage. Inf. Syst.* 34 (4) (2017) 1023–1053, <http://dx.doi.org/10.1080/07421222.2017.1394049>.
- [52] F. Skopik, M. Wurzenberger, M. Landauer, The seven golden principles of effective anomaly-based intrusion detection, *IEEE Secur. Priv.* 19 (5) (2021) 36–45, <http://dx.doi.org/10.1109/MSEC.2021.3090444>.
- [53] Microsoft, Azure digital twins documentation, 2022, Available at: <https://learn.microsoft.com/en-us/azure/digital-twins/>, (Accessed 06 September 2022).
- [54] V. Pavlov, Security aspects of digital twins in IoT platform, 2022, Available at: http://essay.utwente.nl/92695/1/Pavlov_MA_EEMCS.pdf, (Accessed 03 September 2022).
- [55] K. Wüst, A. Gervais, Do you need a blockchain? in: *2018 Crypto Valley Conference on Blockchain Technology*, IEEE, 2018, pp. 45–54, <http://dx.doi.org/10.1109/CVCBT.2018.00011>.
- [56] S.U. Malik, K. Bilal, S.U. Khan, B. Veeravalli, K. Li, A.Y. Zomaya, Modeling and analysis of the thermal properties exhibited by cyberphysical data centers, *IEEE Syst. J.* 11 (1) (2015) 163–172, <http://dx.doi.org/10.1109/JSYST.2015.2493565>.
- [57] S.U. Malik, S.U. Khan, S.K. Srinivasan, Modeling and analysis of state-of-the-art VM-based cloud management platforms, *IEEE Trans. Cloud Comput.* 1 (1) (2013) 1, <http://dx.doi.org/10.1109/TCC.2013.3>.
- [58] L. de Moura, N. Bjørner, Z3: An efficient SMT solver, in: *Tools and Algorithms for the Construction and Analysis of Systems*, Springer Berlin Heidelberg, 2008, pp. 337–340.
- [59] T. Murata, Petri nets: Properties, analysis and applications, *Proc. IEEE* 77 (4) (1989) 541–580, <http://dx.doi.org/10.1109/5.24143>.
- [60] A. Ghosh, et al., Data offloading in IoT environments: modeling, analysis, and verification, *EURASIP J. Wireless Commun. Networking* 2019 (1) (2019) 1–23, <http://dx.doi.org/10.1186/s13638-019-1358-8>.
- [61] M.J.M. Chowdhury, A. Colman, M.A. Kabir, J. Han, P. Sarda, Blockchain versus database: A critical analysis, in: *2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/ 12th IEEE International Conference on Big Data Science and Engineering*, 2018, pp. 1348–1353, <http://dx.doi.org/10.1109/TrustCom/BigDataSE.2018.00186>.
- [62] S. Suhail, M. Iqbal, R. Jurdak, The perils of leveraging evil digital twins as security-enhancing enablers, *Commun. ACM* 67 (1) (2023) 39–42, <http://dx.doi.org/10.1145/3631539>.

- [63] F. Tao, H. Zhang, A. Liu, A.Y. Nee, Digital twin in industry: State-of-the-art, *IEEE Trans. Ind. Inform.* 15 (4) (2018) 2405–2415, <http://dx.doi.org/10.1109/TII.2018.2873186>.
- [64] S. Suhail, M. Iqbal, K. McLaughlin, Digital twin-driven deception platform: Vision and way forward, *IEEE Internet Comput.* (2024) 1–9, <http://dx.doi.org/10.1109/MIC.2024.3406188>.
- [65] S.D. Okegbile, J. Cai, C. Yi, D. Niyato, Human digital twin for personalized healthcare: Vision, architecture and future directions, *IEEE Netw.* (2022) 1–7, <http://dx.doi.org/10.1109/MNET.118.2200071>.
- [66] X. Yang, W. Li, A zero-knowledge-proof-based digital identity management scheme in blockchain, *Comput. Secur.* 99 (2020) 102050, <http://dx.doi.org/10.1016/j.cose.2020.102050>.
- [67] I. Ali, et al., Cognitive digital twins for smart manufacturing, *IEEE Intell. Syst.* 36 (2) (2021) 96–100, <http://dx.doi.org/10.1109/MIS.2021.3062437>.
- [68] M.G. Pieter van Schalkwyk, Sean. Whiteley, Digital twin capabilities periodic table, 2024, Available at: https://www.digitaltwinconsortium.org/wp-content/uploads/sites/3/2024/04/DTC_Capabilities-Periodic-Table-User-Guide-v1.1.pdf, (Accessed 14 April 2024).